training-with-bi-lstm-crf

September 24, 2024

0.1 –

Model Training with Bi-LSTM + CRF for Sequence Labeling

I recently completed training a **Bi-LSTM (Bidirectional Long Short-Term Memory) + CRF (Conditional Random Fields)** model for sequence labeling tasks. This powerful combination enhances the model's ability to capture long-term dependencies and complex relationships in data sequences, such as named entity recognition (NER) and part-of-speech (POS) tagging.

In this project, the Bi-LSTM network effectively captures contextual information in both forward and backward directions, while the CRF layer ensures optimal label predictions by considering the entire sequence structure.

Key takeaways: - Leveraged **Bi-LSTM** for sequential data processing. - Implemented **CRF** to refine and optimize output label sequences. - Achieved impressive accuracy in handling complex sequence-based tasks.

This project adds to my expertise in NLP, deep learning, and advanced model training techniques. Looking forward to sharing more results a## nd insights!

#NLP #DeepLearning #MachineLearning #BiLSTM #CRF share karna chahein.

[54]:
```python
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)


import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/ner-poc-dataset-total-amount-cvr/datasetvalid2.csv
/kaggle/input/ner-only-product-dataset-400/dataset-only-product.csv
/kaggle/input/12-multi-product-712/multi_page.csv
/kaggle/input/01-bccatering-productname-20-july/bc_catering.csv
/kaggle/input/single-product-dataset-10/single_product_dataset.csv
/kaggle/input/inco-03-product-name-26-july/inco_1181.csv
/kaggle/input/dataset-single-productcsv/dataset_single_product.csv
/kaggle/input/ner-project-v2/Data_test_v2.xlsx
/kaggle/input/ner-project-v2/Data_dev.xlsx
/kaggle/input/ner-project-v2/Data_dev_v2.xlsx
```

```
/kaggle/input/ner-project-v2/Data_train.xlsx
/kaggle/input/ner-project-v2/Data_test.xlsx
/kaggle/input/ner-project-v2/Data_train_v2.xlsx
/kaggle/input/02-carlsberg-all-classes/carlsberg_dataset_final.csv
/kaggle/input/06-inco-product-1382-23-aug/1382_inco_dataset.csv
/kaggle/input/inco-1400/1400_inco_split_space.csv
/kaggle/input/04-inco-product-name-11-aug/1400_inco_updated_dataset.csv
/kaggle/input/15-both-pages-ner/merged_file.csv
/kaggle/input/ner-poc-dataset-992/dataset.csv
/kaggle/input/04-carlsberg-all-classes/all_classes_dataset.csv
/kaggle/input/03-carlsberg-all-classes/carlsberg_dataset_updated.csv
/kaggle/input/12-single-page-1292/single_page.csv
/kaggle/input/16-both-pages-only-product-name-ner/Beskrivelse.csv
/kaggle/input/05-inco-product-name-21-aug/1400_inco_updated_dataset.csv
/kaggle/input/kaggleinput17-both-pages-only-product-name-ner/Beskrivelse.csv
/kaggle/input/ner-project/Data_dev.xlsx
/kaggle/input/ner-project/Data_train.xlsx
/kaggle/input/ner-project/Data_test.xlsx
/kaggle/input/11-multi-product-710/dataset.csv
/kaggle/input/ner-all-field-dataset-400/dataset-404.csv
/kaggle/input/13-single-page-1290/13-single-page
/kaggle/input/03-bccatering-product-name-31-july/BC_Catering_dataset.csv
/kaggle/input/inco-03-product-name-21-july/updated_inco_dataset.csv
/kaggle/input/04-bccatering-product-name-8-aug/dataset_bccatering.csv
/kaggle/input/single-product-1290/dataset.csv
/kaggle/input/04-carlsberg-only-products/products_dataset.csv
/kaggle/input/ner-all-field-dataset/dataset-all.csv
/kaggle/input/04-carlsberg-six-classes/six_classes_dataset.csv
/kaggle/input/02-bccatering-26-new-31-july/dataset.csv
/kaggle/input/07-inco-product-name-25-aug/Data_set_inco_vendor_1382.csv
/kaggle/input/carlsberg-485-120623/carlsberg_dataset.csv
/kaggle/input/14-multi-page-711/multi_page_invoices.csv
/kaggle/input/02-inco-product-name-18-july/inco_dataset.csv
/kaggle/input/ner-all-field-200/dataset-200.csv
/kaggle/input/01-inco-product-name-14-july/inco_dataset.csv
```

[55]:
```python
import pandas as pd
import numpy as np
import re
import string
import joblib
import seaborn as sns
import matplotlib.pyplot as plt
```

0.1.1 2. Data pre-processing

[56]:
```
data = pd.read_csv("/kaggle/input/inco-1400/1400_inco_split_space.csv")
```

[57]:
```
# data = data.drop("Unnamed: 3",axis=1)
```

[58]:
```
data.head()
```

[58]:
```
                         Filename   Tokens Tags
0  inv-LSXAP-1669877506.pdf  Faktura    O
1  inv-LSXAP-1669877506.pdf   154551    O
2  inv-LSXAP-1669877506.pdf  Massala    O
3  inv-LSXAP-1669877506.pdf       og    O
4  inv-LSXAP-1669877506.pdf   Hummus    O
```

[59]:
```
# len(set(data['Filename']))
```

[60]:
```
# data_train.rename(columns = {'Tag':'Tag_raw'}, inplace = True)
# data_train.rename(columns = {'Tag_IOB': 'Tag'}, inplace = True)
# data_dev.rename(columns = {'Tag':'Tag_raw'}, inplace = True)
# data_dev.rename(columns = {'Tag_IOB': 'Tag'}, inplace = True)
# data_test.rename(columns = {'Tag':'Tag_raw'}, inplace = True)
# data_test.rename(columns = {'Tag_IOB': 'Tag'}, inplace = True)
# data_train['Tag'] = data_train['Tag'].fillna('O')
# data_dev['Tag'] = data_dev['Tag'].fillna('O')
# data_test['Tag'] = data_test['Tag'].fillna('O')
```

[61]:
```
data.rename(columns = {'Tokens':'token'}, inplace = True)
```

[62]:
```
data.rename(columns = {'Tags':'tag'}, inplace = True)
```

[63]:
```
data.rename(columns = {'Filename':'text'}, inplace = True)
```

[64]:
```
data['tag'] = data['tag'].fillna('O')
```

[65]:
```
len(set(data['text']))
```

[65]: 1400

[66]:
```
data.head()
```

[66]:
```
                             text    token tag
0  inv-LSXAP-1669877506.pdf  Faktura   O
1  inv-LSXAP-1669877506.pdf   154551   O
2  inv-LSXAP-1669877506.pdf  Massala   O
3  inv-LSXAP-1669877506.pdf       og   O
4  inv-LSXAP-1669877506.pdf   Hummus   O
```

2.4. Data processing

[67]:
```
words = list(set(data["token"].values))
words.append("ENDPAD")
n_words = len(words); n_words
```

[67]: 21742

[68]:
```
tags = list(set(data["tag"].values))
n_tags = len(tags); n_tags
```

[68]: 3

[69]:
```
tags
```

[69]: ['O', 'I-BESKRIVELSE', 'B-BESKRIVELSE']

[70]:
```
# data.loc[:, data.isna().any()]
```

[71]:
```
## Concat words in a sentence into a list
class SentenceGetter(object):

    def __init__(self, data):
        self.n_sent = 1
        self.data = data
        self.empty = False
        agg_func = lambda s: [(w, t) for w, t in zip(s["token"].values.tolist(),
                                                      #s["POS"].values.tolist(),
                                                      s["tag"].values.tolist())]
        self.grouped = self.data.groupby("text").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
```

[72]:
```
getter_data = SentenceGetter(data)
```

[73]:
```
getter_data
```

[73]: <__main__.SentenceGetter at 0x78d080253a58>

[74]:
```
sent_train = getter_data.get_next()
# sent_dev = getter_dev.get_next()
```

[75]:
```
sentences_train = getter_data.sentences
# sentences_dev = getter_dev.sentences
```

[76]:
```
data.groupby(['text']).count().max()
```

[76]:
```
token    509
tag      509
dtype: int64
```

[77]:
```
# x = data.groupby('text').count()

# #Mức phân vị thứ 75, 80, 95
# print(x['token'].quantile(q = 0.75))
# print(x['token'].quantile(q = 0.80))
# print(x['token'].quantile(q = 0.95))

# #Plot histogram
# sns.displot(data = t,x='token', kde=True)
# plt.gcf().set_size_inches(30, 20)
```

[78]:
```
max_len = 1000
word2idx = {w: i + 1 for i, w in enumerate(words)}
tag2idx = {t: i for i, t in enumerate(tags)}
idx2tag = {i: w for w, i in tag2idx.items()}

idx2tag
```

[78]:
```
{0: 'O', 1: 'I-BESKRIVELSE', 2: 'B-BESKRIVELSE'}
```

[79]:
```
# word2idx['Massala']
```

[80]:
```
# tag2idx["O"]
```

[81]:
```
from keras.preprocessing.sequence import pad_sequences

# pad the sequence - train sample
X_train = [[word2idx[w[0]] for w in s] for s in sentences_train]

X_train = pad_sequences(maxlen=max_len, sequences=X_train, padding="post",␣
↪value=n_words-1)
# pad the target - dev sample
# y_dev = [[tag2idx[w[1]] for w in s] for s in sentences_dev]
# y_dev = pad_sequences(maxlen=max_len, sequences=y_dev, padding="post",␣
↪value=tag2idx["O"])
```

[82]:
```
# from keras.preprocessing.sequence import pad_sequences
```

[83]:
```
# pad the target - train sample
y_train = [[tag2idx[w[1]] for w in s] for s in sentences_train]
y_train = pad_sequences(maxlen=max_len, sequences=y_train, padding="post")
# y_train = pad_sequences(maxlen=max_len, sequences=y_train, padding="post",␣
 ↪value=tag2idx["O"])
```

[84]:
```
from keras.utils import to_categorical
y_train = [to_categorical(i, num_classes=n_tags) for i in y_train]
# y_dev = [to_categorical(i, num_classes=n_tags) for i in y_dev]

#from sklearn.model_selection import train_test_split
#X_tr, X_te, y_tr, y_te = train_test_split(X, y, test_size=0.1)
```

0.1.2 3. Setup the CRF-LSTM

Now we can fit a LSTM-CRF network with an embedding layer.

[85]:
```
!pip install git+https://www.github.com/keras-team/keras-contrib.git
```

```
Collecting git+https://www.github.com/keras-team/keras-contrib.git
  Cloning https://www.github.com/keras-team/keras-contrib.git to /tmp/pip-req-
build-nosxascv
  Running command git clone -q https://www.github.com/keras-team/keras-
contrib.git /tmp/pip-req-build-nosxascv
Requirement already satisfied (use --upgrade to upgrade): keras-contrib==2.0.8
from git+https://www.github.com/keras-team/keras-contrib.git in
/opt/conda/lib/python3.6/site-packages
Requirement already satisfied: keras in /opt/conda/lib/python3.6/site-packages
(from keras-contrib==2.0.8) (2.3.1)
Requirement already satisfied: h5py in /opt/conda/lib/python3.6/site-packages
(from keras->keras-contrib==2.0.8) (2.9.0)
Requirement already satisfied: numpy>=1.9.1 in /opt/conda/lib/python3.6/site-
packages (from keras->keras-contrib==2.0.8) (1.17.4)
Requirement already satisfied: keras-applications>=1.0.6 in
/opt/conda/lib/python3.6/site-packages (from keras->keras-contrib==2.0.8)
(1.0.8)
Requirement already satisfied: pyyaml in /opt/conda/lib/python3.6/site-packages
(from keras->keras-contrib==2.0.8) (5.1.2)
Requirement already satisfied: scipy>=0.14 in /opt/conda/lib/python3.6/site-
packages (from keras->keras-contrib==2.0.8) (1.3.3)
Requirement already satisfied: keras-preprocessing>=1.0.5 in
/opt/conda/lib/python3.6/site-packages (from keras->keras-contrib==2.0.8)
(1.1.0)
Requirement already satisfied: six>=1.9.0 in /opt/conda/lib/python3.6/site-
packages (from keras->keras-contrib==2.0.8) (1.13.0)
Building wheels for collected packages: keras-contrib
  Building wheel for keras-contrib (setup.py) … done
  Created wheel for keras-contrib: filename=keras_contrib-2.0.8-cp36-none-
```

```
any.whl size=101065
sha256=b06b71e45ccba23f30adb128cba5cda67a3ff34f9ba056e59105a2ff2b901da2
  Stored in directory: /tmp/pip-ephem-wheel-cache-
dsd0iozh/wheels/11/27/c8/4ed56de7b55f4f61244e2dc6ef3cdbaff2692527a2ce6502ba
Successfully built keras-contrib
```

[86]:
```
import keras
import torch
from keras.models import Model, Input,Sequential
from keras.layers import LSTM, Embedding, Dense, TimeDistributed, Dropout,␣
↪Bidirectional
from keras_contrib.layers import CRF
import keras as k
from tensorflow.keras.callbacks import EarlyStopping
```

[87]:
```
import tensorflow
```

[88]:
```
keras.__version__
```

[88]: '2.3.1'

[89]:
```
tensorflow.__version__
```

[89]: '2.1.0-rc0'

[90]:
```
!python3 -V
```

```
Python 3.6.6 :: Anaconda, Inc.
```

[91]:
```
torch.__version__
```

[91]: '1.3.0'

[92]:
```
# !pip freeze > requirements.txt
```

[93]:
```
# !cat requirements.txt
```

[94]:
```
n_words
```

[94]: 21742

[95]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_train, y_train,␣
↪test_size=0.2, random_state=1)
```

[96]:
```
len(X_test)
```

[96]: 280

[97]:
```
len(X_train)
```

[97]: 1120

BI_LSTM_CRF

[98]:
```
import torch
from torch.utils.data import TensorDataset, DataLoader, RandomSampler,␣
 ↪SequentialSampler
# from transformers import BertTokenizer, BertConfig

from keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split

torch.__version__
```

[98]: '1.3.0'

[99]:
```
# device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
# n_gpu = torch.cuda.device_count()
```

[100]:
```
# torch.cuda.get_device_name(0)
```

[101]:
```
# device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
# X_train = torch.tensor(X_train).to(device)
# y_train = torch.tensor(y_train).to(device)
# X_test = torch.tensor(X_test).to(device)
# y_test = torch.tensor(y_test).to(device)
```

[102]:
```
# X_train = torch.tensor(X_train).to(device)
# y_train = torch.tensor(y_train).to(device)
# X_test = torch.tensor(X_test).to(device)
# y_test = torch.tensor(y_test).to(device)
```

[ ]:
```

```

[103]:
```
callback = EarlyStopping(monitor='loss', mode='min', verbose=1, patience=10)#␣
 ↪patience = 50

model = Sequential()
model.add(Embedding(input_dim=n_words+1, output_dim=128, input_length=max_len))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(units=64, return_sequences=True,␣
 ↪recurrent_dropout=0.1)))
model.add(TimeDistributed(Dense(n_tags, activation="relu")))
```

```
crf_layer = CRF(n_tags)
model.add(crf_layer)
```

[ ]:

```
[104]: model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 1000, 128)         2783104
_________________________________________________________________
dropout_2 (Dropout)          (None, 1000, 128)         0
_________________________________________________________________
bidirectional_2 (Bidirection (None, 1000, 128)         98816
_________________________________________________________________
time_distributed_2 (TimeDist (None, 1000, 3)           387
_________________________________________________________________
crf_2 (CRF)                  (None, 1000, 3)           27
=================================================================
Total params: 2,882,334
Trainable params: 2,882,334
Non-trainable params: 0
_________________________________________________________________
```

```
[105]: adam = k.optimizers.Adam(lr=0.0005, beta_1=0.9, beta_2=0.999)
model.compile(optimizer='adam', loss=crf_layer.loss_function,
 metrics=[crf_layer.accuracy])
```

```
/opt/conda/lib/python3.6/site-packages/keras_contrib/layers/crf.py:346:
UserWarning: CRF.loss_function is deprecated and it might be removed in the
future. Please use losses.crf_loss instead.
  warnings.warn('CRF.loss_function is deprecated '
/opt/conda/lib/python3.6/site-packages/keras_contrib/layers/crf.py:353:
UserWarning: CRF.accuracy is deprecated and it might be removed in the future.
Please use metrics.crf_accuracy
  warnings.warn('CRF.accuracy is deprecated and it '
```

```
[106]: history = model.fit(X_train, np.array(y_train), batch_size=32, epochs=200,
                    validation_data=(X_test, np.array(y_test)), verbose=1,
 callbacks=[callback])
```

```
/opt/conda/lib/python3.6/site-
packages/tensorflow_core/python/framework/indexed_slices.py:433: UserWarning:
Converting sparse IndexedSlices to a dense Tensor of unknown shape. This may
consume a large amount of memory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
```

```
Train on 1120 samples, validate on 280 samples
Epoch 1/200
1120/1120 [==============================] - 111s 99ms/step - loss: 0.2160 -
crf_viterbi_accuracy: 0.9751 - val_loss: 0.1101 - val_crf_viterbi_accuracy:
0.9728
Epoch 2/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0849 -
crf_viterbi_accuracy: 0.9751 - val_loss: 0.0695 - val_crf_viterbi_accuracy:
0.9728
Epoch 3/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0584 -
crf_viterbi_accuracy: 0.9751 - val_loss: 0.0540 - val_crf_viterbi_accuracy:
0.9728
Epoch 4/200
1120/1120 [==============================] - 109s 98ms/step - loss: 0.0441 -
crf_viterbi_accuracy: 0.9766 - val_loss: 0.0406 - val_crf_viterbi_accuracy:
0.9770
Epoch 5/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0324 -
crf_viterbi_accuracy: 0.9841 - val_loss: 0.0293 - val_crf_viterbi_accuracy:
0.9871
Epoch 6/200
1120/1120 [==============================] - 110s 98ms/step - loss: 0.0225 -
crf_viterbi_accuracy: 0.9907 - val_loss: 0.0211 - val_crf_viterbi_accuracy:
0.9906
Epoch 7/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0160 -
crf_viterbi_accuracy: 0.9940 - val_loss: 0.0164 - val_crf_viterbi_accuracy:
0.9945
Epoch 8/200
1120/1120 [==============================] - 110s 98ms/step - loss: 0.0124 -
crf_viterbi_accuracy: 0.9965 - val_loss: 0.0140 - val_crf_viterbi_accuracy:
0.9959
Epoch 9/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0102 -
crf_viterbi_accuracy: 0.9978 - val_loss: 0.0123 - val_crf_viterbi_accuracy:
0.9969
Epoch 10/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0087 -
crf_viterbi_accuracy: 0.9986 - val_loss: 0.0110 - val_crf_viterbi_accuracy:
0.9975
Epoch 11/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0078 -
crf_viterbi_accuracy: 0.9990 - val_loss: 0.0103 - val_crf_viterbi_accuracy:
0.9978
Epoch 12/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0072 -
crf_viterbi_accuracy: 0.9993 - val_loss: 0.0100 - val_crf_viterbi_accuracy:
```

```
0.9978
Epoch 13/200
1120/1120 [==============================] - 110s 98ms/step - loss: 0.0067 -
crf_viterbi_accuracy: 0.9995 - val_loss: 0.0097 - val_crf_viterbi_accuracy:
0.9981
Epoch 14/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0063 -
crf_viterbi_accuracy: 0.9996 - val_loss: 0.0096 - val_crf_viterbi_accuracy:
0.9982
Epoch 15/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0060 -
crf_viterbi_accuracy: 0.9996 - val_loss: 0.0092 - val_crf_viterbi_accuracy:
0.9981
Epoch 16/200
1120/1120 [==============================] - 108s 96ms/step - loss: 0.0057 -
crf_viterbi_accuracy: 0.9997 - val_loss: 0.0089 - val_crf_viterbi_accuracy:
0.9983
Epoch 17/200
1120/1120 [==============================] - 109s 98ms/step - loss: 0.0056 -
crf_viterbi_accuracy: 0.9997 - val_loss: 0.0092 - val_crf_viterbi_accuracy:
0.9982
Epoch 18/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0054 -
crf_viterbi_accuracy: 0.9998 - val_loss: 0.0090 - val_crf_viterbi_accuracy:
0.9983
Epoch 19/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0052 -
crf_viterbi_accuracy: 0.9998 - val_loss: 0.0087 - val_crf_viterbi_accuracy:
0.9983
Epoch 20/200
1120/1120 [==============================] - 109s 98ms/step - loss: 0.0051 -
crf_viterbi_accuracy: 0.9998 - val_loss: 0.0087 - val_crf_viterbi_accuracy:
0.9983
Epoch 21/200
1120/1120 [==============================] - 108s 96ms/step - loss: 0.0049 -
crf_viterbi_accuracy: 0.9998 - val_loss: 0.0086 - val_crf_viterbi_accuracy:
0.9984
Epoch 22/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0048 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0084 - val_crf_viterbi_accuracy:
0.9983
Epoch 23/200
1120/1120 [==============================] - 108s 96ms/step - loss: 0.0047 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0087 - val_crf_viterbi_accuracy:
0.9983
Epoch 24/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0046 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0084 - val_crf_viterbi_accuracy:
```

```
0.9984
Epoch 25/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0045 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0082 - val_crf_viterbi_accuracy:
0.9984
Epoch 26/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0044 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0083 - val_crf_viterbi_accuracy:
0.9983
Epoch 27/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0042 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0084 - val_crf_viterbi_accuracy:
0.9984
Epoch 28/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0041 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0082 - val_crf_viterbi_accuracy:
0.9984
Epoch 29/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0040 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0081 - val_crf_viterbi_accuracy:
0.9984
Epoch 30/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0039 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0077 - val_crf_viterbi_accuracy:
0.9984
Epoch 31/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0039 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0078 - val_crf_viterbi_accuracy:
0.9984
Epoch 32/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0038 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0077 - val_crf_viterbi_accuracy:
0.9984
Epoch 33/200
1120/1120 [==============================] - 108s 96ms/step - loss: 0.0037 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0077 - val_crf_viterbi_accuracy:
0.9984
Epoch 34/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0036 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0074 - val_crf_viterbi_accuracy:
0.9984
Epoch 35/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0035 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0076 - val_crf_viterbi_accuracy:
0.9984
Epoch 36/200
1120/1120 [==============================] - 108s 96ms/step - loss: 0.0034 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0075 - val_crf_viterbi_accuracy:
```

```
0.9984
Epoch 37/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0034 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0074 - val_crf_viterbi_accuracy:
0.9984
Epoch 38/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0033 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0073 - val_crf_viterbi_accuracy:
0.9984
Epoch 39/200
1120/1120 [==============================] - 110s 98ms/step - loss: 0.0032 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0073 - val_crf_viterbi_accuracy:
0.9984
Epoch 40/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0032 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0070 - val_crf_viterbi_accuracy:
0.9985
Epoch 41/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0031 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0071 - val_crf_viterbi_accuracy:
0.9985
Epoch 42/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0030 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0073 - val_crf_viterbi_accuracy:
0.9984
Epoch 43/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0030 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0071 - val_crf_viterbi_accuracy:
0.9985
Epoch 44/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0029 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0068 - val_crf_viterbi_accuracy:
0.9985
Epoch 45/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0028 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0071 - val_crf_viterbi_accuracy:
0.9984
Epoch 46/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0028 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0066 - val_crf_viterbi_accuracy:
0.9985
Epoch 47/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0027 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0071 - val_crf_viterbi_accuracy:
0.9985
Epoch 48/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0027 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0070 - val_crf_viterbi_accuracy:
```

```
0.9984
Epoch 49/200
1120/1120 [==============================] - 108s 96ms/step - loss: 0.0026 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0070 - val_crf_viterbi_accuracy:
0.9984
Epoch 50/200
1120/1120 [==============================] - 108s 96ms/step - loss: 0.0026 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0065 - val_crf_viterbi_accuracy:
0.9986
Epoch 51/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0025 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0067 - val_crf_viterbi_accuracy:
0.9985
Epoch 52/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0025 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0063 - val_crf_viterbi_accuracy:
0.9985
Epoch 53/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0024 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0066 - val_crf_viterbi_accuracy:
0.9986
Epoch 54/200
1120/1120 [==============================] - 108s 96ms/step - loss: 0.0024 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0065 - val_crf_viterbi_accuracy:
0.9985
Epoch 55/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0023 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0067 - val_crf_viterbi_accuracy:
0.9985
Epoch 56/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0023 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0065 - val_crf_viterbi_accuracy:
0.9985
Epoch 57/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0022 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0063 - val_crf_viterbi_accuracy:
0.9986
Epoch 58/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0022 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0064 - val_crf_viterbi_accuracy:
0.9985
Epoch 59/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0022 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0062 - val_crf_viterbi_accuracy:
0.9986
Epoch 60/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0021 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0065 - val_crf_viterbi_accuracy:
```

```
0.9985
Epoch 61/200
1120/1120 [==============================] - 108s 96ms/step - loss: 0.0021 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0065 - val_crf_viterbi_accuracy:
0.9985
Epoch 62/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0021 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0065 - val_crf_viterbi_accuracy:
0.9985
Epoch 63/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0020 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0065 - val_crf_viterbi_accuracy:
0.9986
Epoch 64/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0020 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0063 - val_crf_viterbi_accuracy:
0.9985
Epoch 65/200
1120/1120 [==============================] - 109s 98ms/step - loss: 0.0020 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0061 - val_crf_viterbi_accuracy:
0.9986
Epoch 66/200
1120/1120 [==============================] - 108s 96ms/step - loss: 0.0019 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0062 - val_crf_viterbi_accuracy:
0.9986
Epoch 67/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0019 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0061 - val_crf_viterbi_accuracy:
0.9985
Epoch 68/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0019 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0065 - val_crf_viterbi_accuracy:
0.9985
Epoch 69/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0019 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0059 - val_crf_viterbi_accuracy:
0.9986
Epoch 70/200
1120/1120 [==============================] - 110s 98ms/step - loss: 0.0018 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0060 - val_crf_viterbi_accuracy:
0.9986
Epoch 71/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0018 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0059 - val_crf_viterbi_accuracy:
0.9986
Epoch 72/200
1120/1120 [==============================] - 109s 98ms/step - loss: 0.0018 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0060 - val_crf_viterbi_accuracy:
```

```
0.9986
Epoch 73/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0017 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0060 - val_crf_viterbi_accuracy:
0.9986
Epoch 74/200
1120/1120 [==============================] - 109s 98ms/step - loss: 0.0017 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0061 - val_crf_viterbi_accuracy:
0.9986
Epoch 75/200
1120/1120 [==============================] - 108s 96ms/step - loss: 0.0017 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0061 - val_crf_viterbi_accuracy:
0.9986
Epoch 76/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0017 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0059 - val_crf_viterbi_accuracy:
0.9986
Epoch 77/200
1120/1120 [==============================] - 111s 99ms/step - loss: 0.0016 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0062 - val_crf_viterbi_accuracy:
0.9986
Epoch 78/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0016 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0060 - val_crf_viterbi_accuracy:
0.9986
Epoch 79/200
1120/1120 [==============================] - 109s 98ms/step - loss: 0.0016 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0061 - val_crf_viterbi_accuracy:
0.9986
Epoch 80/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0016 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0061 - val_crf_viterbi_accuracy:
0.9986
Epoch 81/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0015 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0060 - val_crf_viterbi_accuracy:
0.9986
Epoch 82/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0015 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0058 - val_crf_viterbi_accuracy:
0.9987
Epoch 83/200
1120/1120 [==============================] - 109s 98ms/step - loss: 0.0015 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0059 - val_crf_viterbi_accuracy:
0.9986
Epoch 84/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0015 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0060 - val_crf_viterbi_accuracy:
```

```
0.9986
Epoch 85/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0015 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0057 - val_crf_viterbi_accuracy:
0.9987
Epoch 86/200
1120/1120 [==============================] - 109s 98ms/step - loss: 0.0015 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0058 - val_crf_viterbi_accuracy:
0.9986
Epoch 87/200
1120/1120 [==============================] - 108s 96ms/step - loss: 0.0014 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0057 - val_crf_viterbi_accuracy:
0.9986
Epoch 88/200
1120/1120 [==============================] - 109s 98ms/step - loss: 0.0014 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0059 - val_crf_viterbi_accuracy:
0.9986
Epoch 89/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0014 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0060 - val_crf_viterbi_accuracy:
0.9985
Epoch 90/200
1120/1120 [==============================] - 109s 98ms/step - loss: 0.0014 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0061 - val_crf_viterbi_accuracy:
0.9986
Epoch 91/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0014 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0056 - val_crf_viterbi_accuracy:
0.9987
Epoch 92/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0013 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0059 - val_crf_viterbi_accuracy:
0.9986
Epoch 93/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0013 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0056 - val_crf_viterbi_accuracy:
0.9987
Epoch 94/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0013 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0057 - val_crf_viterbi_accuracy:
0.9987
Epoch 95/200
1120/1120 [==============================] - 109s 98ms/step - loss: 0.0013 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0060 - val_crf_viterbi_accuracy:
0.9986
Epoch 96/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0013 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0057 - val_crf_viterbi_accuracy:
```

```
0.9987
Epoch 97/200
1120/1120 [==============================] - 110s 98ms/step - loss: 0.0013 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0058 - val_crf_viterbi_accuracy:
0.9986
Epoch 98/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0013 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0057 - val_crf_viterbi_accuracy:
0.9987
Epoch 99/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0012 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0061 - val_crf_viterbi_accuracy:
0.9985
Epoch 100/200
1120/1120 [==============================] - 109s 98ms/step - loss: 0.0012 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0064 - val_crf_viterbi_accuracy:
0.9985
Epoch 101/200
1120/1120 [==============================] - 108s 96ms/step - loss: 0.0012 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0056 - val_crf_viterbi_accuracy:
0.9987
Epoch 102/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0012 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0057 - val_crf_viterbi_accuracy:
0.9986
Epoch 103/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0012 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0057 - val_crf_viterbi_accuracy:
0.9986
Epoch 104/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0012 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0055 - val_crf_viterbi_accuracy:
0.9986
Epoch 105/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0012 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0056 - val_crf_viterbi_accuracy:
0.9987
Epoch 106/200
1120/1120 [==============================] - 108s 96ms/step - loss: 0.0011 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0055 - val_crf_viterbi_accuracy:
0.9986
Epoch 107/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0011 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0054 - val_crf_viterbi_accuracy:
0.9987
Epoch 108/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0011 -
crf_viterbi_accuracy: 0.9999 - val_loss: 0.0053 - val_crf_viterbi_accuracy:
```

```
0.9987
Epoch 109/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0011 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0056 - val_crf_viterbi_accuracy:
0.9987
Epoch 110/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0011 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0054 - val_crf_viterbi_accuracy:
0.9987
Epoch 111/200
1120/1120 [==============================] - 110s 98ms/step - loss: 0.0011 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0056 - val_crf_viterbi_accuracy:
0.9987
Epoch 112/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0011 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0053 - val_crf_viterbi_accuracy:
0.9987
Epoch 113/200
1120/1120 [==============================] - 108s 96ms/step - loss: 0.0010 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0059 - val_crf_viterbi_accuracy:
0.9987
Epoch 114/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0010 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0053 - val_crf_viterbi_accuracy:
0.9987
Epoch 115/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0010 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0055 - val_crf_viterbi_accuracy:
0.9987
Epoch 116/200
1120/1120 [==============================] - 108s 97ms/step - loss: 0.0010 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0056 - val_crf_viterbi_accuracy:
0.9987
Epoch 117/200
1120/1120 [==============================] - 109s 97ms/step - loss: 0.0010 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0054 - val_crf_viterbi_accuracy:
0.9987
Epoch 118/200
1120/1120 [==============================] - 108s 96ms/step - loss: 9.8806e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0058 - val_crf_viterbi_accuracy:
0.9987
Epoch 119/200
1120/1120 [==============================] - 109s 97ms/step - loss: 9.7471e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0057 - val_crf_viterbi_accuracy:
0.9987
Epoch 120/200
1120/1120 [==============================] - 110s 98ms/step - loss: 9.6931e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0056 - val_crf_viterbi_accuracy:
```

```
0.9987
Epoch 121/200
1120/1120 [==============================] - 111s 99ms/step - loss: 9.6520e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0055 - val_crf_viterbi_accuracy:
0.9987
Epoch 122/200
1120/1120 [==============================] - 109s 97ms/step - loss: 9.4448e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0056 - val_crf_viterbi_accuracy:
0.9987
Epoch 123/200
1120/1120 [==============================] - 110s 98ms/step - loss: 9.3907e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0055 - val_crf_viterbi_accuracy:
0.9987
Epoch 124/200
1120/1120 [==============================] - 109s 98ms/step - loss: 9.2525e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0057 - val_crf_viterbi_accuracy:
0.9987
Epoch 125/200
1120/1120 [==============================] - 111s 99ms/step - loss: 9.1620e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0057 - val_crf_viterbi_accuracy:
0.9987
Epoch 126/200
1120/1120 [==============================] - 108s 97ms/step - loss: 9.0610e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0057 - val_crf_viterbi_accuracy:
0.9987
Epoch 127/200
1120/1120 [==============================] - 109s 97ms/step - loss: 9.0188e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0056 - val_crf_viterbi_accuracy:
0.9987
Epoch 128/200
1120/1120 [==============================] - 108s 97ms/step - loss: 8.8545e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0056 - val_crf_viterbi_accuracy:
0.9987
Epoch 129/200
1120/1120 [==============================] - 108s 97ms/step - loss: 8.6380e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0055 - val_crf_viterbi_accuracy:
0.9987
Epoch 130/200
1120/1120 [==============================] - 110s 98ms/step - loss: 8.7892e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0055 - val_crf_viterbi_accuracy:
0.9987
Epoch 131/200
1120/1120 [==============================] - 108s 97ms/step - loss: 8.5172e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0054 - val_crf_viterbi_accuracy:
0.9987
Epoch 132/200
1120/1120 [==============================] - 109s 98ms/step - loss: 8.5356e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0060 - val_crf_viterbi_accuracy:
```

```
0.9987
Epoch 133/200
1120/1120 [==============================] - 108s 97ms/step - loss: 8.2967e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0052 - val_crf_viterbi_accuracy:
0.9987
Epoch 134/200
1120/1120 [==============================] - 109s 97ms/step - loss: 8.1961e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0055 - val_crf_viterbi_accuracy:
0.9987
Epoch 135/200
1120/1120 [==============================] - 108s 96ms/step - loss: 8.0797e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0058 - val_crf_viterbi_accuracy:
0.9987
Epoch 136/200
1120/1120 [==============================] - 108s 97ms/step - loss: 8.0423e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0056 - val_crf_viterbi_accuracy:
0.9987
Epoch 137/200
1120/1120 [==============================] - 109s 97ms/step - loss: 7.9071e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0053 - val_crf_viterbi_accuracy:
0.9987
Epoch 138/200
1120/1120 [==============================] - 108s 96ms/step - loss: 7.7891e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0058 - val_crf_viterbi_accuracy:
0.9987
Epoch 139/200
1120/1120 [==============================] - 108s 97ms/step - loss: 7.6510e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0057 - val_crf_viterbi_accuracy:
0.9987
Epoch 140/200
1120/1120 [==============================] - 108s 96ms/step - loss: 7.6273e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0050 - val_crf_viterbi_accuracy:
0.9987
Epoch 141/200
1120/1120 [==============================] - 108s 96ms/step - loss: 7.4925e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0055 - val_crf_viterbi_accuracy:
0.9987
Epoch 142/200
1120/1120 [==============================] - 109s 97ms/step - loss: 7.3057e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0055 - val_crf_viterbi_accuracy:
0.9987
Epoch 143/200
1120/1120 [==============================] - 108s 97ms/step - loss: 7.2095e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0052 - val_crf_viterbi_accuracy:
0.9987
Epoch 144/200
1120/1120 [==============================] - 109s 97ms/step - loss: 7.1061e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0056 - val_crf_viterbi_accuracy:
```

```
0.9987
Epoch 145/200
1120/1120 [==============================] - 108s 97ms/step - loss: 7.0249e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0052 - val_crf_viterbi_accuracy:
0.9987
Epoch 146/200
1120/1120 [==============================] - 109s 97ms/step - loss: 6.9862e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0052 - val_crf_viterbi_accuracy:
0.9987
Epoch 147/200
1120/1120 [==============================] - 108s 96ms/step - loss: 6.8303e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0053 - val_crf_viterbi_accuracy:
0.9987
Epoch 148/200
1120/1120 [==============================] - 108s 96ms/step - loss: 6.6617e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0058 - val_crf_viterbi_accuracy:
0.9986
Epoch 149/200
1120/1120 [==============================] - 109s 97ms/step - loss: 6.6640e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0057 - val_crf_viterbi_accuracy:
0.9987
Epoch 150/200
1120/1120 [==============================] - 107s 96ms/step - loss: 6.4548e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0053 - val_crf_viterbi_accuracy:
0.9987
Epoch 151/200
1120/1120 [==============================] - 109s 97ms/step - loss: 6.4018e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0054 - val_crf_viterbi_accuracy:
0.9987
Epoch 152/200
1120/1120 [==============================] - 107s 96ms/step - loss: 6.2874e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0052 - val_crf_viterbi_accuracy:
0.9988
Epoch 153/200
1120/1120 [==============================] - 108s 97ms/step - loss: 6.1618e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0055 - val_crf_viterbi_accuracy:
0.9986
Epoch 154/200
1120/1120 [==============================] - 110s 98ms/step - loss: 6.0394e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0053 - val_crf_viterbi_accuracy:
0.9987
Epoch 155/200
1120/1120 [==============================] - 109s 97ms/step - loss: 5.9703e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0052 - val_crf_viterbi_accuracy:
0.9988
Epoch 156/200
1120/1120 [==============================] - 108s 97ms/step - loss: 5.8732e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0053 - val_crf_viterbi_accuracy:
```

```
0.9987
Epoch 157/200
1120/1120 [==============================] - 109s 97ms/step - loss: 5.7379e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0053 - val_crf_viterbi_accuracy:
0.9987
Epoch 158/200
1120/1120 [==============================] - 108s 97ms/step - loss: 5.6642e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0050 - val_crf_viterbi_accuracy:
0.9988
Epoch 159/200
1120/1120 [==============================] - 107s 96ms/step - loss: 5.6190e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0049 - val_crf_viterbi_accuracy:
0.9988
Epoch 160/200
1120/1120 [==============================] - 107s 96ms/step - loss: 5.2927e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0052 - val_crf_viterbi_accuracy:
0.9988
Epoch 161/200
1120/1120 [==============================] - 108s 96ms/step - loss: 5.3430e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0051 - val_crf_viterbi_accuracy:
0.9988
Epoch 162/200
1120/1120 [==============================] - 108s 96ms/step - loss: 5.1313e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0054 - val_crf_viterbi_accuracy:
0.9987
Epoch 163/200
1120/1120 [==============================] - 108s 97ms/step - loss: 5.0568e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0055 - val_crf_viterbi_accuracy:
0.9987
Epoch 164/200
1120/1120 [==============================] - 108s 96ms/step - loss: 4.8828e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0050 - val_crf_viterbi_accuracy:
0.9988
Epoch 165/200
1120/1120 [==============================] - 107s 96ms/step - loss: 4.7949e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0053 - val_crf_viterbi_accuracy:
0.9988
Epoch 166/200
1120/1120 [==============================] - 108s 97ms/step - loss: 4.7047e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0048 - val_crf_viterbi_accuracy:
0.9988
Epoch 167/200
1120/1120 [==============================] - 107s 96ms/step - loss: 4.5434e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0054 - val_crf_viterbi_accuracy:
0.9987
Epoch 168/200
1120/1120 [==============================] - 108s 97ms/step - loss: 4.3581e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0052 - val_crf_viterbi_accuracy:
```

```
0.9987
Epoch 169/200
1120/1120 [==============================] - 108s 96ms/step - loss: 4.2602e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0053 - val_crf_viterbi_accuracy:
0.9988
Epoch 170/200
1120/1120 [==============================] - 107s 96ms/step - loss: 4.2625e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0051 - val_crf_viterbi_accuracy:
0.9987
Epoch 171/200
1120/1120 [==============================] - 108s 97ms/step - loss: 4.0347e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0055 - val_crf_viterbi_accuracy:
0.9987
Epoch 172/200
1120/1120 [==============================] - 108s 96ms/step - loss: 3.9708e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0053 - val_crf_viterbi_accuracy:
0.9987
Epoch 173/200
1120/1120 [==============================] - 108s 97ms/step - loss: 3.8483e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0072 - val_crf_viterbi_accuracy:
0.9982
Epoch 174/200
1120/1120 [==============================] - 107s 96ms/step - loss: 4.1650e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0057 - val_crf_viterbi_accuracy:
0.9985
Epoch 175/200
1120/1120 [==============================] - 107s 96ms/step - loss: 3.6610e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0046 - val_crf_viterbi_accuracy:
0.9988
Epoch 176/200
1120/1120 [==============================] - 108s 96ms/step - loss: 3.4398e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0053 - val_crf_viterbi_accuracy:
0.9987
Epoch 177/200
1120/1120 [==============================] - 107s 96ms/step - loss: 3.3201e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0051 - val_crf_viterbi_accuracy:
0.9988
Epoch 178/200
1120/1120 [==============================] - 108s 97ms/step - loss: 3.2939e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0052 - val_crf_viterbi_accuracy:
0.9987
Epoch 179/200
1120/1120 [==============================] - 107s 96ms/step - loss: 3.0398e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0052 - val_crf_viterbi_accuracy:
0.9987
Epoch 180/200
1120/1120 [==============================] - 107s 96ms/step - loss: 2.8951e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0052 - val_crf_viterbi_accuracy:
```

```
0.9987
Epoch 181/200
1120/1120 [==============================] - 109s 97ms/step - loss: 2.7751e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0052 - val_crf_viterbi_accuracy:
0.9988
Epoch 182/200
1120/1120 [==============================] - 107s 96ms/step - loss: 2.5877e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0052 - val_crf_viterbi_accuracy:
0.9987
Epoch 183/200
1120/1120 [==============================] - 107s 96ms/step - loss: 2.4956e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0053 - val_crf_viterbi_accuracy:
0.9987
Epoch 184/200
1120/1120 [==============================] - 108s 96ms/step - loss: 2.3919e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0052 - val_crf_viterbi_accuracy:
0.9987
Epoch 185/200
1120/1120 [==============================] - 107s 96ms/step - loss: 2.1610e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0053 - val_crf_viterbi_accuracy:
0.9987
Epoch 186/200
1120/1120 [==============================] - 108s 96ms/step - loss: 2.1147e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0055 - val_crf_viterbi_accuracy:
0.9987
Epoch 187/200
1120/1120 [==============================] - 107s 96ms/step - loss: 2.1048e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0053 - val_crf_viterbi_accuracy:
0.9987
Epoch 188/200
1120/1120 [==============================] - 107s 96ms/step - loss: 1.7735e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0049 - val_crf_viterbi_accuracy:
0.9988
Epoch 189/200
1120/1120 [==============================] - 108s 96ms/step - loss: 1.5703e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0050 - val_crf_viterbi_accuracy:
0.9987
Epoch 190/200
1120/1120 [==============================] - 107s 96ms/step - loss: 1.4401e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0054 - val_crf_viterbi_accuracy:
0.9987
Epoch 191/200
1120/1120 [==============================] - 108s 96ms/step - loss: 1.3423e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0054 - val_crf_viterbi_accuracy:
0.9987
Epoch 192/200
1120/1120 [==============================] - 107s 96ms/step - loss: 1.1406e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0053 - val_crf_viterbi_accuracy:
```

```
0.9987
Epoch 193/200
1120/1120 [==============================] - 108s 97ms/step - loss: 1.0385e-04 -
crf_viterbi_accuracy: 1.0000 - val_loss: 0.0052 - val_crf_viterbi_accuracy:
0.9987
Epoch 194/200
1120/1120 [==============================] - 119s 106ms/step - loss: 9.1027e-05
- crf_viterbi_accuracy: 1.0000 - val_loss: 0.0049 - val_crf_viterbi_accuracy:
0.9988
Epoch 195/200
1120/1120 [==============================] - 121s 108ms/step - loss: 6.6648e-05
- crf_viterbi_accuracy: 1.0000 - val_loss: 0.0051 - val_crf_viterbi_accuracy:
0.9988
Epoch 196/200
1120/1120 [==============================] - 126s 113ms/step - loss: 6.2246e-05
- crf_viterbi_accuracy: 1.0000 - val_loss: 0.0050 - val_crf_viterbi_accuracy:
0.9987
Epoch 197/200
1120/1120 [==============================] - 128s 114ms/step - loss: 4.0519e-05
- crf_viterbi_accuracy: 1.0000 - val_loss: 0.0048 - val_crf_viterbi_accuracy:
0.9988
Epoch 198/200
1120/1120 [==============================] - 129s 115ms/step - loss: 2.1421e-05
- crf_viterbi_accuracy: 1.0000 - val_loss: 0.0049 - val_crf_viterbi_accuracy:
0.9987
Epoch 199/200
1120/1120 [==============================] - 129s 115ms/step - loss: 1.0765e-05
- crf_viterbi_accuracy: 1.0000 - val_loss: 0.0049 - val_crf_viterbi_accuracy:
0.9988
Epoch 200/200
1120/1120 [==============================] - 131s 117ms/step - loss: -1.0311e-05
- crf_viterbi_accuracy: 1.0000 - val_loss: 0.0046 - val_crf_viterbi_accuracy:
0.9988
```

[107]:
```
3+4
```

[107]: 7

[108]:
```
# import torch
# import torch.nn as nn
# import torch.optim as optim
```

[109]:
```
# device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
# X_train = torch.tensor(X_train).to(device)
# y_train = torch.tensor(y_train).to(device)
# X_test = torch.tensor(X_test).to(device)
# y_test = torch.tensor(y_test).to(device)
```

[110]:
```
# model = Sequential()
# model.add(Embedding(input_dim=n_words+1, output_dim=128,␣
↪input_length=max_len))
# model.add(Dropout(0.5))
# model.add(Bidirectional(LSTM(units=64, return_sequences=True,␣
↪recurrent_dropout=0.1)))
# model.add(TimeDistributed(Dense(n_tags, activation="relu")))
# crf_layer = CRF(n_tags)
# model.add(crf_layer)
# # model.to(device)
```

[111]:
```
# # Move the model to GPU
# device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
# # model.to(device)

# # Define your optimizer
# parameters = []
# for name, module in model.named_modules():
#     if isinstance(module, nn.Module):
#         parameters.extend(module.named_parameters())
# optimizer = optim.Adam(parameters, lr=0.0005, betas=(0.9, 0.999))
```

[112]:
```
# history = []
# for epoch in range(50):
#     model.train()
#     optimizer.zero_grad()
#     outputs = model(X_train)
#     loss = crf_layer.loss_function(outputs, y_train)
#     loss.backward()
#     optimizer.step()
#     history.append(loss.item())

#     model.eval()
#     with torch.no_grad():
#         val_outputs = model(X_test)
#         val_loss = crf_layer.loss_function(val_outputs, y_test)

#     print(f"Epoch {epoch+1}/{50}, Loss: {loss.item():.4f}, Val Loss:␣
↪{val_loss.item():.4f}")

#     # Add early stopping logic here if needed

# # Additional code for evaluation and prediction if required
```

[ ]:
```

```

[113]:
```
model.save("model.h5")
joblib.dump(words, 'words.pkl')
joblib.dump(tags, 'tags.pkl')
```

[113]: ['tags.pkl']

[114]:
```
!pip install zip
```

```
Collecting zip
  Downloading https://files.pythonhosted.org/packages/dd/31/1c0dc71cd947a5c48f18
b0ff9d8fd3a0da0bad9fa63c36dfd9715676926d/zip-0.0.2.tar.gz
Collecting Flask-Admin>=1.0.4
  Downloading https://files.pythonhosted.org/packages/61/b3/656c78dfef1635
17dbbc9fd106f0604e37b436ad51f9d9450b60e9407e35/Flask_Admin-1.6.1-py3-none-
any.whl (7.5MB)
     |                  | 7.5MB 3.4MB/s eta 0:00:01
|                  | 6.9MB 3.4MB/s eta 0:00:01
Collecting Flask-Bootstrap>=2.2.2-1
  Downloading https://files.pythonhosted.org/packages/88/53/958ce7c2aa2628
0b7fd7f3eecbf13053f1302ee2acb1db58ef32e1c23c2a/Flask-Bootstrap-3.3.7.1.tar.gz
(456kB)
     |                  | 460kB 37.1MB/s eta 0:00:01
Collecting Flask-Cache>=0.10.1
  Downloading https://files.pythonhosted.org/packages/91/c4/f71095437bd4b6
91c63f240e72a20c57e2c216085cbc271f79665885d3da/Flask-Cache-0.13.1.tar.gz (45kB)
     |                  | 51kB 4.6MB/s  eta 0:00:01
Collecting Flask-FlatPages>=0.3
  Downloading https://files.pythonhosted.org/packages/ed/9b/d86fa78a07bb72bfe13c
018979dc31a73ad4dd725a8a764d600b4f46ddee/Flask_FlatPages-0.8.1-py2.py3-none-
any.whl
Collecting Flask-Gravatar>=0.2.4
  Downloading https://files.pythonhosted.org/packages/58/4a/b20260e8d383d0037f27
91dd8a3f3ea729ca9f02d7638677a34a236a8702/Flask_Gravatar-0.5.0-py2.py3-none-
any.whl
Collecting Flask-Login>=0.1.3
  Downloading https://files.pythonhosted.org/packages/2b/83/ac5bf3279f969704fc1e
63f050c50e10985e50fd340e6069ec7e09df5442/Flask_Login-0.5.0-py2.py3-none-any.whl
Collecting Flask-Mail>=0.7.4
  Downloading https://files.pythonhosted.org/packages/05/2f/6a545452040c25
56559779db87148d2a85e78a26f90326647b51dc5e81e9/Flask-Mail-0.9.1.tar.gz (45kB)
     |                  | 51kB 4.9MB/s  eta 0:00:01
Collecting Flask-PyMongo>=0.2.1
  Downloading https://files.pythonhosted.org/packages/67/b8/0322016b9ce09a64fba9
018211e7c35fd51380527ffd9ea248744f389239/Flask_PyMongo-2.3.0-py2.py3-none-
any.whl
Collecting Flask-Restless>=0.9.1
  Downloading https://files.pythonhosted.org/packages/ae/ad/14eee74ef110f2
```

```
bd8641de98675037f037dd06d614f7c435671be66a55c7/Flask-Restless-0.17.0.tar.gz
(42kB)
     |                 | 51kB 4.4MB/s  eta 0:00:01
Collecting Flask-SQLAlchemy>=0.16
  Downloading https://files.pythonhosted.org/packages/26/2c/9088b6bd95bca539230b
be9ad446737ed391aab9a83aff403e18dded3e75/Flask_SQLAlchemy-2.5.1-py2.py3-none-
any.whl
Collecting Flask-Themes>=0.1.3
  Downloading https://files.pythonhosted.org/packages/1a/12/5be3cc2a56c63277177a
7e3db4a8f346b9f68a6641277bb3029d401688e3/Flask-Themes-0.1.3.tar.gz
Collecting Flask-Uploads>=0.1.3
  Downloading https://files.pythonhosted.org/packages/c9/a8/2c8e9ec04267d94b7852
a374cebeb9a32d60f8cba83818af960e64fafbec/Flask-Uploads-0.2.1.tar.gz
Collecting Flask-WTF>=0.8.2
  Downloading https://files.pythonhosted.org/packages/3a/26/3803ee692eb9a8d21bf7
ba1cecd649ce3a55899c65467bdfc1bad13ec50f/Flask_WTF-1.0.1-py3-none-any.whl
Requirement already satisfied: Flask>=0.9 in /opt/conda/lib/python3.6/site-
packages (from zip) (1.1.1)
Collecting frozen-flask
  Downloading https://files.pythonhosted.org/packages/42/02/251de2bf6dfa5dbc2ac5
5e0cb63d8b48a6e9e08bfc44bcced47560529081/Frozen_Flask-0.18-py3-none-any.whl
Requirement already satisfied: Jinja2>=2.6 in /opt/conda/lib/python3.6/site-
packages (from zip) (2.10.3)
Requirement already satisfied: Markdown>=2.2.1 in /opt/conda/lib/python3.6/site-
packages (from zip) (3.1.1)
Requirement already satisfied: PyYAML>=3.11 in /opt/conda/lib/python3.6/site-
packages (from zip) (5.1.2)
Requirement already satisfied: SQLAlchemy>=0.8.0b2 in
/opt/conda/lib/python3.6/site-packages (from zip) (1.3.11)
Requirement already satisfied: Sphinx>=1.3.1 in /opt/conda/lib/python3.6/site-
packages (from zip) (2.2.1)
Collecting WTForms>=1.0.3
  Downloading https://files.pythonhosted.org/packages/9a/38/58698b4bfbc0e9
3200af1fbe886cc6eb1ff31232b9224b4ebc12356e2f18/WTForms-3.0.0-py3-none-any.whl
(136kB)
     |                 | 143kB 28.3MB/s eta 0:00:01
Requirement already satisfied: Werkzeug>=0.8.3 in
/opt/conda/lib/python3.6/site-packages (from zip) (0.16.0)
Collecting argparse>=1.2.1
  Downloading https://files.pythonhosted.org/packages/f2/94/3af39d34be01a24a6e65
433d19e107099374224905f1e0cc6bbe1fd22a2f/argparse-1.4.0-py2.py3-none-any.whl
Collecting blinker>=1.2
  Downloading https://files.pythonhosted.org/packages/30/41/caa5da2dbe6d26029dfe
11d31dfa8132b4d6d30b6d6b61a24824075a5f06/blinker-1.5-py2.py3-none-any.whl
Collecting bumpversion>=0.5.3
  Downloading https://files.pythonhosted.org/packages/4e/ff/93f0db7b3ca337e9f2a2
89980083e858775dfb3672b38052c6911b36ea66/bumpversion-0.6.0-py2.py3-none-any.whl
Requirement already satisfied: click>=6.3 in /opt/conda/lib/python3.6/site-
```

```
packages (from zip) (7.0)
Requirement already satisfied: colorama>=0.3.7 in /opt/conda/lib/python3.6/site-
packages (from zip) (0.4.1)
Requirement already satisfied: coverage>=4.0 in /opt/conda/lib/python3.6/site-
packages (from zip) (4.5.4)
Requirement already satisfied: cryptography>=1.0.1 in
/opt/conda/lib/python3.6/site-packages (from zip) (2.3.1)
Requirement already satisfied: flake8>=2.4.1 in /opt/conda/lib/python3.6/site-
packages (from zip) (3.6.0)
Requirement already satisfied: networkx>=1.11 in /opt/conda/lib/python3.6/site-
packages (from zip) (2.4)
Requirement already satisfied: pymongo>=2.5.1 in /opt/conda/lib/python3.6/site-
packages (from zip) (3.9.0)
Requirement already satisfied: pytest>=2.8.3 in /opt/conda/lib/python3.6/site-
packages (from zip) (5.0.1)
Requirement already satisfied: python-dateutil>=1.5 in
/opt/conda/lib/python3.6/site-packages (from zip) (2.8.0)
Requirement already satisfied: six>=1.10.0 in /opt/conda/lib/python3.6/site-
packages (from zip) (1.13.0)
Collecting tox>=2.1.1
  Downloading https://files.pythonhosted.org/packages/31/c7/69ec9b8bbb4fe6
49b4bc960a53b837b4dace889137be5f23b4236a3e9f7c/tox-3.28.0-py2.py3-none-any.whl
(86kB)
     |                 | 92kB 7.1MB/s  eta 0:00:01
Collecting watchdog>=0.8.3
  Downloading https://files.pythonhosted.org/packages/79/21/ffd41427b724a6
468c6c5c7f083f8e59948eabe2538538e3a15ff15c33cb/watchdog-2.3.1-py3-none-
manylinux2014_x86_64.whl (80kB)
     |                 | 81kB 6.6MB/s  eta 0:00:01
Requirement already satisfied: wheel>=0.23.0 in
/opt/conda/lib/python3.6/site-packages (from zip) (0.33.6)
Collecting wsgiref>=0.1.2
  Downloading https://files.pythonhosted.org/packages/41/9e/309259ce8dff8c596e8c
26df86dbc4e848b9249fd36797fd60be456f03fc/wsgiref-0.1.2.zip
```

```
    ERROR: Command errored out with exit status 1:
     command: /opt/conda/bin/python -c 'import sys, setuptools, tokenize;
sys.argv[0] = '"'"'/tmp/pip-install-wzbemok1/wsgiref/setup.py'"'"';
__file__='"'"'/tmp/pip-install-
wzbemok1/wsgiref/setup.py'"'"';f=getattr(tokenize, '"'"'open'"'"',
open)(__file__);code=f.read().replace('"'"'\r\n'"'"',
'"'"'\n'"'"');f.close();exec(compile(code, __file__, '"'"'exec'"'"'))' egg_info
--egg-base /tmp/pip-install-wzbemok1/wsgiref/pip-egg-info
         cwd: /tmp/pip-install-wzbemok1/wsgiref/
    Complete output (8 lines):
    Traceback (most recent call last):
      File "<string>", line 1, in <module>
      File "/tmp/pip-install-wzbemok1/wsgiref/setup.py", line 5, in <module>
        import ez_setup
      File "/tmp/pip-install-wzbemok1/wsgiref/ez_setup/__init__.py", line 170
        print "Setuptools version",version,"or greater has been installed."
                                 ^
    SyntaxError: Missing parentheses in call to 'print'. Did you mean
print("Setuptools version",version,"or greater has been installed.")?
    ----------------------------------------
ERROR: Command errored out with exit status 1: python setup.py egg_info
Check the logs for full command output.
```

[115]:

```python
import zipfile
import os
from IPython.display import FileLink

def zip_dir(directory = os.curdir, file_name = 'directory.zip'):
    """
    zip all the files in a directory

    Parameters
    -----
    directory: str
        directory needs to be zipped, defualt is current working directory

    file_name: str
```

```
        the name of the zipped file (including .zip), default is 'directory.zip'

    Returns
    -----
    Creates a hyperlink, which can be used to download the zip file)
    """
    os.chdir(directory)
    zip_ref = zipfile.ZipFile(file_name, mode='w')
    for folder, _, files in os.walk(directory):
        for file in files:
            if file_name in file:
                pass
            else:
                zip_ref.write(os.path.join(folder, file))

    return FileLink(file_name)
```

[116]:
```
zip_dir()
```

[116]: /kaggle/working/directory.zip

[ ]:
```

```

[117]:
```
# np.shape(y_train)
```

[118]:
```
# from keras.wrappers.scikit_learn import KerasClassifier
# from sklearn.model_selection import GridSearchCV

# # define a function that returns the Keras model
# def create_model(lr=0.0005, lstm_units=64):
#     early_stopping = EarlyStopping(monitor='val_loss', mode='min', verbose=1,␣
↪patience=50)
#     model = Sequential()
#     model.add(Embedding(input_dim=n_words+1, output_dim=128,␣
↪input_length=max_len))
#     model.add(Dropout(0.5))
#     model.add(Bidirectional(LSTM(units=lstm_units, return_sequences=True,␣
↪recurrent_dropout=0.1)))
#     model.add(TimeDistributed(Dense(n_tags, activation="relu")))
#     crf_layer = CRF(n_tags)
#     model.add(crf_layer)

#     adam = k.optimizers.Adam(lr=lr, beta_1=0.9, beta_2=0.999)
#     model.compile(optimizer='adam', loss=crf_layer.loss_function,␣
↪metrics=[crf_layer.accuracy])
#     return model
```

```
# # create the Keras model
# model = KerasClassifier(build_fn=create_model, epochs=100, batch_size=256,␣
↪verbose=1)


# # define the grid search parameters
# # learning_rates = [0.001, 0.0005, 0.0001]
# # lstm_units = [32, 64, 128]
# # param_grid = dict(lr=learning_rates, lstm_units=lstm_units)

# # perform the grid search
# # grid = GridSearchCV(estimator=model, param_grid=param_grid, cv=5,␣
↪scoring='accuracy')
# grid_result = model.fit(X_train, y_train)
```

[119]:
```
# np.shape(X_train)
```

[120]:
```
# np.shape(y_train)
```

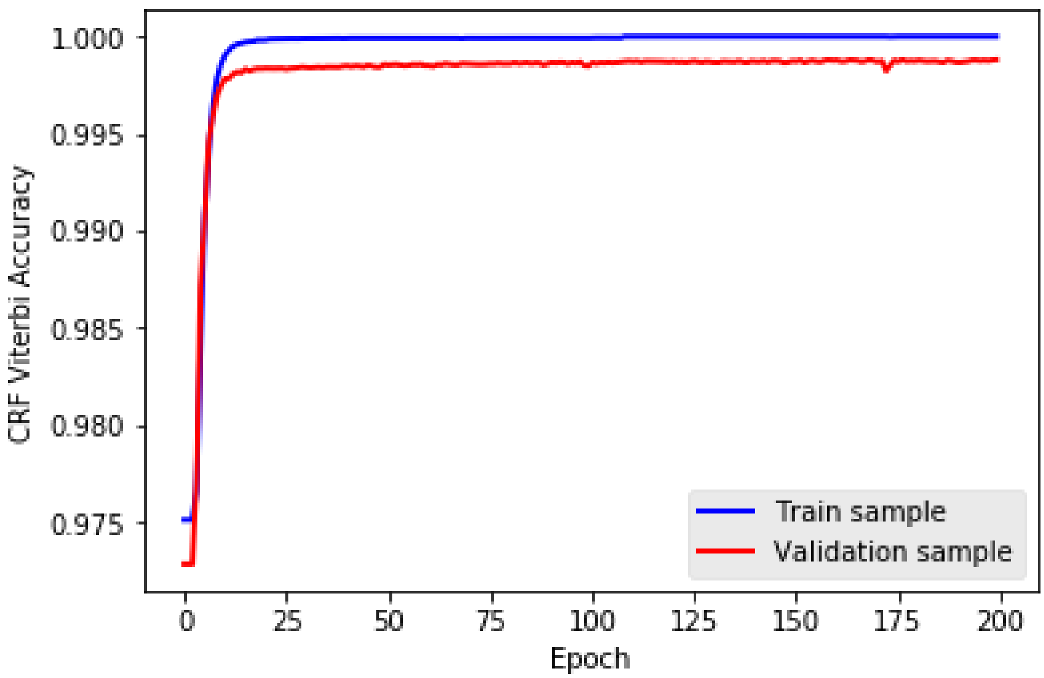
[121]:
```
# %%bash
# pip install transformers==4.1.1
```

[122]:
```
# import numpy as np
# import pandas as pd
# import tensorflow as tf
# import transformers
# from transformers import BertTokenizer, TFBertModel

# # Load the BERT tokenizer and model
# tokenizer = BertTokenizer.from_pretrained('bert-base-cased')
# model = TFBertModel.from_pretrained('bert-base-cased')

# # Define the input and output layers of the model
# input_ids = tf.keras.layers.Input(shape=(128,), dtype=tf.int32,␣
↪name='input_ids')
# attention_mask = tf.keras.layers.Input(shape=(128,), dtype=tf.int32,␣
↪name='attention_mask')
# outputs = model({'input_ids': input_ids, 'attention_mask': attention_mask})[0]
# outputs = tf.keras.layers.Dropout(0.1)(outputs)
# outputs = tf.keras.layers.Dense(7, activation='softmax')(outputs)

# # Define the model
# model = tf.keras.Model(inputs=[input_ids, attention_mask], outputs=outputs)

# # Compile the model
# optimizer = tf.keras.optimizers.Adam(lr=5e-5)
# loss = tf.keras.losses.SparseCategoricalCrossentropy()
```

```
# accuracy = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')
# model.compile(optimizer=optimizer, loss=loss, metrics=[accuracy])


# history = model.fit(X_train, np.array(y_train), epochs=100, batch_size=256,␣
 ↪verbose=1,validation_split=0.1)
```

[123]:
```
hist = pd.DataFrame(history.history)
```

[124]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
plt.style.use("ggplot")
plt.figure(figsize=(12,12))
l1, = ax.plot(hist["crf_viterbi_accuracy"], label = 'Train sample',␣
 ↪color='blue', linewidth = 2)
l2, = ax.plot(hist["val_crf_viterbi_accuracy"], label = 'Validation␣
 ↪sample',color='red', linewidth = 2)
ax.set(xlabel='Epoch', ylabel='CRF Viterbi Accuracy')
ax.legend((l1,l2),('Train sample','Validation sample'))
plt.show()
```




```
<Figure size 864x864 with 0 Axes>
```

[125]:
```
!pip install seqeval
```

```
Collecting seqeval
  Downloading https://files.pythonhosted.org/packages/9d/2d/233c79d5b4e5ab
1dbf111242299153f3caddddbb691219f363ad55ce783d/seqeval-1.2.2.tar.gz (43kB)
     |                 | 51kB 2.5MB/s eta 0:00:011
Requirement already satisfied: numpy>=1.14.0 in
/opt/conda/lib/python3.6/site-packages (from seqeval) (1.17.4)
Requirement already satisfied: scikit-learn>=0.21.3 in
/opt/conda/lib/python3.6/site-packages (from seqeval) (0.21.3)
Requirement already satisfied: scipy>=0.17.0 in /opt/conda/lib/python3.6/site-
packages (from scikit-learn>=0.21.3->seqeval) (1.3.3)
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.6/site-
packages (from scikit-learn>=0.21.3->seqeval) (0.14.0)
Building wheels for collected packages: seqeval
  Building wheel for seqeval (setup.py) … done
  Created wheel for seqeval: filename=seqeval-1.2.2-cp36-none-any.whl
size=16172
sha256=2d5a636dff08d9f88c236e9aaaf015e2d78571f74fd7b143fd3ea89a9e0b5fdc
  Stored in directory: /root/.cache/pip/wheels/52/df/1b/45d75646c37428f7e6262147
04a0e35bd3cfc32eda37e59e5f
Successfully built seqeval
Installing collected packages: seqeval
Successfully installed seqeval-1.2.2
```

[126]:
```
from seqeval.metrics import precision_score, recall_score, f1_score,␣
↪classification_report
```

[127]:
```
train_pred = model.predict(X_train, verbose=1)
```

```
1120/1120 [==============================] - 26s 24ms/step
```

[128]:
```
test_pred = model.predict(X_test, verbose=1)
```

```
280/280 [==============================] - 6s 23ms/step
```

[129]:
```
# def pred2label(pred):
#     out = []
#     for pred_i in pred:
#         out_i = []
#         for p in pred_i:
#             p_i = np.argmax(p)
#             tag = idx2tag[p_i]
#             if isinstance(tag, float):
#                 tag = str(tag)
#             out_i.append(tag.replace("PAD", "O").replace("nan", "O"))
#         out.append(out_i)
#     return out
```

[130]:
```python
def pred2label(pred):
    out = []
    for pred_i in pred:
        out_i = []
        for p in pred_i:
            p_i = np.argmax(p)
            out_i.append(idx2tag[p_i].replace("PAD", "O"))
        out.append(out_i)
    return out
```

[131]:
```python
train_pred_labels = pred2label(train_pred)
train_actual_labels = pred2label(y_train)

test_pred_labels = pred2label(test_pred)
test_actual_labels = pred2label(y_test)
```

[132]:
```python
print("F1-score: {:.1%}".format(f1_score(train_actual_labels,␣
 ↪train_pred_labels)))
print("F1-score: {:.1%}".format(f1_score(test_actual_labels, test_pred_labels)))
```

```
F1-score: 99.9%
F1-score: 91.1%
```

[133]:
```python
print(classification_report(train_actual_labels, train_pred_labels))
```

```
              precision    recall  f1-score   support

 BESKRIVELSE       1.00      1.00      1.00      7617

   micro avg       1.00      1.00      1.00      7617
   macro avg       1.00      1.00      1.00      7617
weighted avg       1.00      1.00      1.00      7617
```

[134]:
```python
print(classification_report(test_actual_labels, test_pred_labels))
```

```
              precision    recall  f1-score   support

 BESKRIVELSE       0.92      0.91      0.91      2053

   micro avg       0.92      0.91      0.91      2053
   macro avg       0.92      0.91      0.91      2053
weighted avg       0.92      0.91      0.91      2053
```

[135]:
```python
model.evaluate(X_test, np.array(y_test))
```

```
280/280 [==============================] - 9s 31ms/step
```

[135]: [0.004580638018835868, 0.9988043904304504]

[136]:
```python
# p = model.predict(np.array([X_test[i]]))
# p = np.argmax(p, axis=-1)
# true = np.argmax(y_test[i], -1)
# print("{},{},{}".format("Word", "True", "Pred"))
# for w, t, pred in zip(X_test[i], true, p[0]):
# #     if w != 0:
#     print("{},{},{}".format(words[w-1], tags[t], tags[pred]))
```

[137]:
```python
i = 1
p = model.predict(np.array([X_test[i]]))
p = np.argmax(p, axis=-1)
true = np.argmax(y_test[i], -1)
print("{},{},{}".format("Word", "True", "Pred"))
for w, t, pred in zip(X_test[i], true, p[0]):
#     if w != 0:
    print("{},{},{}".format(words[w-1], tags[t], tags[pred]))
```

```
Word,True,Pred
Faktura,0,0
Åbningstider,0,0
Mandag,0,0
-,0,0
fredag:,0,0
Lørdag:,0,0
Søndag:,0,0
6:00,0,0
-,0,0
17:00,0,0
8:00,0,0
-,0,0
14:00,0,0
8:00,0,0
-,0,0
14:00,0,0
inco,0,0
CC,0,0
Glostrup,0,0
A/S,0,0
Ejby,0,0
Industrivej,0,0
111,,0,0
DK,0,0
2600,0,0
Glostrup,0,0
Tlf:,0,0
```

72108181,0,0
Fax:,0,0
621391,0,0
Gorms,0,0
Food,0,0
&,0,0
Catering,0,0
ApS,0,0
S,0,0
Att.,0,0
Christian,0,0
Madsen,0,0
Marielundvej,0,0
34,0,0
A,0,0
2730,0,0
Herlev,0,0
Kortholder:,0,0
Jack,0,0
Christensen,0,0
Økologikontrolmyndighed:,0,0
DK-ØKO-100,0,0
Bank,0,0
896,0,0
1039038,0,0
CVR-nr.,0,0
37589608,0,0
Side,0,0
1,0,0
Fakturanr.,0,0
Fakturadato,0,0
Fakturatidspunkt,0,0
Kundenr.,0,0
Kunde,0,0
CVR-nr.,0,0
Betales,0,0
inden,0,0
Terminal,0,0
Bonnr.,0,0
302,0,0
Eksp.,0,0
96020017,0,0
3302321766,0,0
14,0,0
juni,0,0
2022,0,0
08:19:48,0,0
621391,0,0

29816271,0,0
15,0,0
juni,0,0
2022,0,0
4,0,0
302000326265,0,0
Kortnr.,0,0
Varenr.,0,0
Antal,0,0
Enhed,0,0
Antal,0,0
pr.,0,0
enhed,0,0
Beskrivelse,0,0
Listepris,0,0
excl.,0,0
moms,0,0
Enhedspris,0,0
Beløb,0,0
201414,0,0
114931,0,0
1,0,0
1,0,0
PS,0,0
STK,0,0
1,0,0
PS,0,0
1,0,0
STK,0,0
GRØNT,0,0
Rødløg,B-BESKRIVELSE,B-BESKRIVELSE
hele,I-BESKRIVELSE,I-BESKRIVELSE
skrællet,I-BESKRIVELSE,I-BESKRIVELSE
5kg,I-BESKRIVELSE,I-BESKRIVELSE
Tomater,B-BESKRIVELSE,B-BESKRIVELSE
stilk,I-BESKRIVELSE,I-BESKRIVELSE
5kg,I-BESKRIVELSE,I-BESKRIVELSE
Oprindelse,0,0
på,0,0
frugt,0,0
og,0,0
grønt,0,0
fremgår,0,0
af,0,0
varen,,0,0
hvis,0,0
det,0,0
ikke,0,0

```
fremgår,0,0
af,0,0
fakturaen,0,0
Subtotal,0,0
.,0,0
.,0,0
.,0,0
.,0,0
.,0,0
.,0,0
.,0,0
.,0,0
74,99,0,0
54,99,0,0
37,50,0,0
54,99,0,0
92,49,0,0
37,50,0,0
54,99,0,0
Ordreliniertotal,0,0
Total,0,0
før,0,0
moms,0,0
Total,0,0
moms,0,0
25%,0,0
Fakturabeløb,0,0
92,49,0,0
92,49,0,0
23,12,0,0
115,61,0,0
Rabat,0,0
ialt,0,0
denne,0,0
faktura,0,0
37,50,0,0
71<,0,0
330232176603,0,0
+87063453<,0,0
Beløbet,0,0
trækkes,0,0
gennem,0,0
PBS-Leverandørservice,0,0
på,0,0
forfaldsdagen,0,0
Efter,0,0
forfald,0,0
renteberegnes,0,0
```

```
med,0,0
2,0,0
%,0,0
pr.,0,0
måned,0,0
=,0,0
fast,0,0
lavpris,0,0
-,0,0
ø,0,0
=,0,0
økologisk,0,0
vare,0,0
inco,0,0
CC,0,0
Glostrup,0,0
A/S,0,0
•,0,0
Ejby,0,0
Industrivej,0,0
111,0,0
•,0,0
2600,0,0
Glostrup,0,0
•,0,0
Tlf.:,0,0
45,0,0
72,0,0
10,0,0
81,0,0
81,0,0
•,0,0
inco@inco.dk,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
```

```
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
```

```
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
```

```
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
```

```
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
```

```
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
```

```
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
```

```
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
```

```
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
```

```
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
```

```
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
```

```
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
```

```
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
```

```
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
```

```
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
```

```
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
```

```
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
```

```
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
10:32:44,0,0
```

[138]:
```
# p
```

[139]:
```
# Custom Tokenizer
re_tok = re.compile(f'([{string.punctuation}""¨«»®´·º½¾¿¡§£₤‘’])')
def tokenize(s): return s.split()
```

[140]:
```
test_sentence=''
x_test_sent = pad_sequences(sequences=[[word2idx.get(w, 0) for w in␣
 ↪tokenize(test_sentence)]],
                            padding="post", value=0, maxlen=max_len)
p = model.predict(np.array([x_test_sent[0]]))
p = np.argmax(p, axis=-1)
print("{:15}||{}".format("Word", "Prediction"))
print(30 * "=")
for w, pred in zip(tokenize(test_sentence), p[0]):
    print("{:15}: {:5}".format(w, tags[pred]))
```

```
Word           ||Prediction
==============================
```

[141]:
```
len(data)
```

[141]: 384608

[142]:
```
# get sentences
# data_test
getter_test = SentenceGetter(data)
sent_test = getter_test.get_next()
sentences_test = getter_test.sentences
```

[143]:
```
# pad the sequence - test sample
X_test = [[word2idx[w[0]] for w in s] for s in sentences_test]
```

```
X_test = pad_sequences(maxlen=max_len, sequences=X_test, padding="post",␣
↪value=n_words-1)

# pad the target - dev sample
y_test = [[tag2idx[w[1]] for w in s] for s in sentences_test]
# y_test = pad_sequences(maxlen=max_len, sequences=y_test, padding="post",␣
↪value=tag2idx["O"])
y_test = pad_sequences(maxlen=max_len, sequences=y_test, padding="post")

y_test = [to_categorical(i, num_classes=n_tags) for i in y_test]
```

[144]:
```
test_pred = model.predict(X_test, verbose=1)
```

```
1400/1400 [==============================] - 31s 22ms/step
```

[145]:
```
test_pred_labels = pred2label(test_pred)
test_actual_labels = pred2label(y_test)
```

[146]:
```
print("F1-score: {:.1%}".format(f1_score(test_actual_labels, test_pred_labels)))
```

```
F1-score: 98.1%
```

[147]:
```
print(classification_report(test_actual_labels, test_pred_labels))
```

```
              precision    recall  f1-score   support

 BESKRIVELSE       0.98      0.98      0.98      9670

   micro avg       0.98      0.98      0.98      9670
   macro avg       0.98      0.98      0.98      9670
weighted avg       0.98      0.98      0.98      9670
```

[ ]:
```

```

0.2 Example

[148]:
```
# d = {'Index_': [1,2,3,4,5,6,7,8,9,10,11,12],
#      'NER': ['PERSONTYPE',
#              'PERSON',
#              'ADDRESS',
#              'PHONENUMBER',
#              'DATETIME',
#              'ORGANIZATION',
#              'LOCATION',
#              'PRODUCT',
#              'EMAIL',
```

```
#               'MISCELLANEOUS',
#               'URL',
#               'IP'],
#       'Sentence' : ['Tổng giám đốc đi công tác tại Phú Yên.',
#                     'Nguyễn Văn Thanh được phong làm NSND.',
#                     'Chiếc điện thoại này cần giao tới địa chỉ số 57, đường Lý␣
↪Thường Kiệt, p.Trần Hưng Đạo, Hà Nội.',
#                     'Hãy gọi cho cô Hoa vào số điện thoại 0934.456.787.',
#                     '15 giờ sáng ngày 24 tháng 6 năm 2021, xảy ra một vụ cháy␣
↪tại quận Thanh Xuân.',
#                     'Tòa án nhân dân tối cao là cơ quan xét xử cao nhất của␣
↪nước Cộng hòa xã hội chủ nghĩa Việt Nam.',
#                     'Đông Nam Á là tiểu vùng địa lý phía đông nam của châu Á.',
#                     'Iphone 14 là phiên bản điện thoại mới nhất của tập đoàn␣
↪công nghệ Apple.',
#                     'Email của tôi là hanoi@gmail.com.',
#                     'COVID-19 là một bệnh đường hô hấp cấp tính truyền nhiễm␣
↪gây ra bởi chủng virus corona SARS-CoV-2 và các biến thể của nó.',
#                     'Bạn có thể đọc tin tức tại trang vnexpress.net.',
#                     'Máy tính có IP là 120.126.1.1 hiện đang bị lỗi.']}
# data_sample = pd.DataFrame(d)
```

[149]:
```
# data_sample
```

[150]:
```
# def test_sentence(index):
#     test_sentence=d['Sentence'][index]
#     x_test_sent = pad_sequences(sequences=[[word2idx.get(w, 0) for w in␣
↪tokenize(test_sentence)]],padding="post", value=0, maxlen=max_len)
#     p = model.predict(np.array([x_test_sent[0]]))
#     p = np.argmax(p, axis=-1)
#     print("{:15}||{}".format("Word", "Prediction"))
#     print(30 * "=")
#     for w, pred in zip(tokenize(test_sentence), p[0]):
#         print("{:15}: {:5}".format(w, tags[pred]))
```

[ ]:
```

```

[151]:
```
def test_sentence_sample(test_sentence):
    #test_sentence="Anh Mai đang bị hâm"
    x_test_sent = pad_sequences(sequences=[[word2idx.get(w, 0) for w in␣
↪tokenize(test_sentence)]],padding="post", value=0, maxlen=max_len)
    p = model.predict(np.array([x_test_sent[0]]))
    p = np.argmax(p, axis=-1)
    print("{:15}||{}".format("Word", "Prediction"))
    print(30 * "=")
    for w, pred in zip(tokenize(test_sentence), p[0]):
```

```
        print("{:15}: {:5}".format(w, tags[pred]))
```

[152]:
```
test_sentence_sample("Frostvarer 201204 Jordbær 1 Total FRIGODAN 1 PS (2,5 KG)")
```

```
Word           ||Prediction
==============================
Frostvarer     : I-BESKRIVELSE
201204         : B-BESKRIVELSE
Jordbær        : I-BESKRIVELSE
1              : O
Total          : O
FRIGODAN       : O
1              : O
PS             : O
(2,5           : O
KG)            : O
```

[153]:
```
!pip install  gdown
```

```
Collecting gdown
  Downloading https://files.pythonhosted.org/packages/e7/38/e3393edb5fd157abaa54
292f31251f8c2ff739673f535990f8a43e69b9dd/gdown-4.7.1-py3-none-any.whl
Requirement already satisfied: tqdm in /opt/conda/lib/python3.6/site-packages
(from gdown) (4.39.0)
Requirement already satisfied: beautifulsoup4 in /opt/conda/lib/python3.6/site-
packages (from gdown) (4.8.1)
Requirement already satisfied: requests[socks] in /opt/conda/lib/python3.6/site-
packages (from gdown) (2.22.0)
Requirement already satisfied: filelock in /opt/conda/lib/python3.6/site-
packages (from gdown) (3.0.12)
Requirement already satisfied: six in /opt/conda/lib/python3.6/site-packages
(from gdown) (1.13.0)
Requirement already satisfied: soupsieve>=1.2 in /opt/conda/lib/python3.6/site-
packages (from beautifulsoup4->gdown) (1.9.5)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/opt/conda/lib/python3.6/site-packages (from requests[socks]->gdown) (1.24.2)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in
/opt/conda/lib/python3.6/site-packages (from requests[socks]->gdown) (3.0.4)
Requirement already satisfied: idna<2.9,>=2.5 in /opt/conda/lib/python3.6/site-
packages (from requests[socks]->gdown) (2.8)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.6/site-packages (from requests[socks]->gdown) (2019.9.11)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6; extra == "socks" in
/opt/conda/lib/python3.6/site-packages (from requests[socks]->gdown) (1.7.1)
Installing collected packages: gdown
Successfully installed gdown-4.7.1
```

[154]:
```
!gdown https://drive.google.com/file/d/1Y3jPwcGvO-bndkEKNNTYu4vJHreroezj/
  ↪view?usp=sharing --fuzzy
```

```
Downloading…
From: https://drive.google.com/uc?id=1Y3jPwcGvO-bndkEKNNTYu4vJHreroezj
To: /kaggle/working/inv-Tl91T-1630501455.pdf
100%|                    | 91.6k/91.6k [00:00<00:00, 44.1MB/s]
```

[155]:
```
conda install -c conda-forge pdfminer.six
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.7.12
  latest version: 23.7.3

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: /opt/conda

  added / updated specs:
    - pdfminer.six


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2023.7.22  |       hbcca054_0         146 KB  conda-forge
    certifi-2020.6.20          |     pyhd3eb1b0_3         155 KB
    openssl-1.0.2u             |       h516909a_0         3.2 MB  conda-forge
    pdfminer.six-20201018      |     pyhd8ed1ab_3         4.9 MB  conda-forge
    ------------------------------------------------------------
                                           Total:         8.3 MB

The following NEW packages will be INSTALLED:

  pdfminer.six       conda-forge/noarch::pdfminer.six-20201018-pyhd8ed1ab_3

The following packages will be UPDATED:
```

```
  ca-certificates    pkgs/main::ca-certificates-2019.10.16~ --> conda-forge::ca-
certificates-2023.7.22-hbcca054_0
  certifi            pkgs/main/linux-64::certifi-2019.9.11~ -->
pkgs/main/noarch::certifi-2020.6.20-pyhd3eb1b0_3
  openssl             pkgs/main::openssl-1.0.2t-h7b6447c_1 --> conda-
forge::openssl-1.0.2u-h516909a_0



Downloading and Extracting Packages
certifi-2020.6.20    | 155 KB    | ##################################### | 100%
ca-certificates-2023 | 146 KB    | ##################################### | 100%
openssl-1.0.2u       | 3.2 MB    | ##################################### | 100%
pdfminer.six-2020101 | 4.9 MB    | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

[156]:
```python
import csv
from pdfminer.high_level import extract_text

pdf_file = "/kaggle/working/inv-T191T-1630501455.pdf"
csv_file = "/kaggle/working/output.csv"

# Extract text from pdf
text = extract_text(pdf_file)
# Generate tokens
tokens = text.split()

print(tokens)

check =  ' '.join(tokens)
print(check)
```

```
['Totalleverandør', 'af', 'fødevarer', 'og', 'nonfood-artikler', 'House', 'of',
'Shawarma', 'Fantastic', 'Food', 'Aps', 'Rosengårdcentret', 'Ørbækvej', '75',
'5220', 'Odense', 'SØ', 'Kundenr.', '.', '.', '.', '.', '.', '.', '.', '.', '.',
'79874', 'Økologikontrolmyndighed:', 'DK-ØKO-100', 'Kopi', '*10-3162056*',
'Faktura', '.', '.', '.', '.', '.', '.', 'Bogføringsdato', '.', '.', '.', '.',
'.', '.', '.', '.', '.', '.', '.', 'Rute', 'Sælger', 'Betalingsbetingels', '.',
'.', '.', '.', '.', '.', 'Betales', 'inden', '.', '.', '.', '.', '.', '.', '.',
'.', '.', '.', '.', 'Side', '10-3162056', '09-07-21', '55', 'CGOPHA/CGOCLG',
'30', 'Dage', 'Netto', '08-08-21', '1', '59007', 'Nummer', 'Beskrivelse',
'Mærke', 'Antal', 'Enhed', 'Salgspris', 'Beløb', '206873', 'Agurk', 'm/Film',
'(NL)', '206928', 'Koriander', 'i', 'Bundt', '(KE)', '206863', 'Løg', 'i', 'PS',
```

'(DK)', '229956', 'Persille', 'Bredbladet', 'i', 'Bundt', '(IT)', '206826', 'Salat', 'Iceberg', '(NL)', '218441', 'Tomat', 'Udenlandsk', '57-67', '(NL)', '67418', 'Yoghurt', 'Græsk', 'Inspireret', '10%', '3232', 'Yoghurt', 'Sødmælk', 'Naturel', '3,5%', '2435', 'Salatmayonnaise', '50%', '42893', 'Salatmayonnaise', 'Portion', '42891', 'Tomatketchup', 'Portionspakning', '72647', 'Sødmælk', 'UHT', '3,5%', '316302', 'Fritureolie', '316304', 'Rapsolie', '21390', 'Eddike', 'u/Konserveringsmid', '5%', 'Kølevarer', 'GRØNT', 'KRYDDER', 'HØJVANG', 'KRYDDER', 'GRØNT', 'GRØNT', 'SOL', 'ARLA', 'PRO', 'KRAFT', 'BÄHNCKE', 'BÄHNCKE', 'GOURMET', 'DELTA', 'DELTA', 'LAGERBE', 'Tørvarer', 'Frostvarer', '3', '2', '2', '2', '8', '4', '1', '10', '1', '3', '3', '12', '6', '2', '4', 'STK', 'PS', 'PS', 'BT', 'KRT', 'KRT', 'SP', '(5', 'KG)', 'SP', '(5', 'LTR)', 'SP', '(10', 'KG)', 'ÆSK', '(126x25', 'G)', 'ÆSK', '(126x25', 'G)', 'STK', '(1', 'LTR)', 'DK', '(10', 'LTR)', 'DK', '(10', 'LTR)', 'DK', '(10', 'LTR)', '4,74', '11,96', '6,35', '10,31', '65,40', '67,62', '83,35', '60,45', '149,00', '125,50', '129,05', '7,75', '108,00', '108,00', '39,50', '14,22', '23,92', '12,70', '20,62', '523,20', '270,48', '83,35', '604,50', '149,00', '376,50', '387,15', '93,00', '648,00', '216,00', '158,00', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '*', '234734', 'PommesFrites', '7x7', 'mm', 'FARMFRIT', '12', 'KRT', '(5x2,5', 'KG)', '106,25', '1.275,00', 'Moms', 'nr.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '39902060', '+71<', '000010316205607', '+89136296<', 'I', 'alt', 'DKK', '25%', 'moms', 'I', 'alt', 'DKK', 'inkl.', 'moms', '4.855,64', '1.213,91', '6.069,55', 'BC', 'Catering', 'Grossisten', 'A/S', 'Blækhatten', '10', '5220', 'Odense', 'SØ', 'Tlf.', '63', '15', '88', '55', 'CVR', '10', '83', '70', '49', 'www.bccatering.dk']
Totalleverandør af fødevarer og nonfood-artikler House of Shawarma Fantastic Food Aps Rosengårdcentret Ørbækvej 75 5220 Odense SØ Kundenr. . . . . 79874 Økologikontrolmyndighed: DK-ØKO-100 Kopi *10-3162056* Faktura . . . . . . Bogføringsdato . . . . Rute Sælger Betalingsbetingels . . . . . . Betales inden . . . . Side 10-3162056 09-07-21 55 CGOPHA/CGOCLG 30 Dage Netto 08-08-21 1 59007 Nummer Beskrivelse Mærke Antal Enhed Salgspris Beløb 206873 Agurk m/Film (NL) 206928 Koriander i Bundt (KE) 206863 Løg i PS (DK) 229956 Persille Bredbladet i Bundt (IT) 206826 Salat Iceberg (NL) 218441 Tomat Udenlandsk 57-67 (NL) 67418 Yoghurt Græsk Inspireret 10% 3232 Yoghurt Sødmælk Naturel 3,5% 2435 Salatmayonnaise 50% 42893 Salatmayonnaise Portion 42891 Tomatketchup Portionspakning 72647 Sødmælk UHT 3,5% 316302 Fritureolie 316304 Rapsolie 21390 Eddike u/Konserveringsmid 5% Kølevarer GRØNT KRYDDER HØJVANG KRYDDER GRØNT GRØNT SOL ARLA PRO KRAFT BÄHNCKE BÄHNCKE GOURMET DELTA DELTA LAGERBE Tørvarer Frostvarer 3 2 2 2 8 4 1 10 1 3 3 12 6 2 4 STK PS PS BT KRT KRT SP (5 KG) SP (5 LTR) SP (10 KG) ÆSK (126x25 G) ÆSK (126x25 G) STK (1 LTR) DK (10 LTR) DK (10 LTR) DK (10 LTR) 4,74 11,96 6,35 10,31 65,40 67,62 83,35 60,45 149,00 125,50 129,05 7,75 108,00 108,00 39,50 14,22 23,92 12,70 20,62 523,20 270,48 83,35 604,50 149,00 376,50 387,15 93,00 648,00 216,00 158,00 * * * * * * * * * * * * * * * * 234734 PommesFrites 7x7 mm FARMFRIT 12 KRT (5x2,5 KG) 106,25 1.275,00 Moms nr. . . . . 39902060 +71< 000010316205607 +89136296< I alt DKK 25% moms I alt DKK inkl. moms 4.855,64 1.213,91 6.069,55 BC Catering Grossisten A/S Blækhatten 10 5220 Odense SØ Tlf. 63 15 88 55 CVR 10 83 70 49 www.bccatering.dk

[157]:
```
!pip install git+https://www.github.com/keras-team/keras-contrib.git
```

```
Collecting git+https://www.github.com/keras-team/keras-contrib.git
  Cloning https://www.github.com/keras-team/keras-contrib.git to /tmp/pip-req-
build-78lflzgz
  Running command git clone -q https://www.github.com/keras-team/keras-
contrib.git /tmp/pip-req-build-78lflzgz
Requirement already satisfied (use --upgrade to upgrade): keras-contrib==2.0.8
from git+https://www.github.com/keras-team/keras-contrib.git in
/opt/conda/lib/python3.6/site-packages
Requirement already satisfied: keras in /opt/conda/lib/python3.6/site-packages
(from keras-contrib==2.0.8) (2.3.1)
Requirement already satisfied: scipy>=0.14 in /opt/conda/lib/python3.6/site-
packages (from keras->keras-contrib==2.0.8) (1.3.3)
Requirement already satisfied: numpy>=1.9.1 in /opt/conda/lib/python3.6/site-
packages (from keras->keras-contrib==2.0.8) (1.17.4)
Requirement already satisfied: h5py in /opt/conda/lib/python3.6/site-packages
(from keras->keras-contrib==2.0.8) (2.9.0)
Requirement already satisfied: keras-preprocessing>=1.0.5 in
/opt/conda/lib/python3.6/site-packages (from keras->keras-contrib==2.0.8)
(1.1.0)
Requirement already satisfied: six>=1.9.0 in /opt/conda/lib/python3.6/site-
packages (from keras->keras-contrib==2.0.8) (1.13.0)
Requirement already satisfied: keras-applications>=1.0.6 in
/opt/conda/lib/python3.6/site-packages (from keras->keras-contrib==2.0.8)
(1.0.8)
Requirement already satisfied: pyyaml in /opt/conda/lib/python3.6/site-packages
(from keras->keras-contrib==2.0.8) (5.1.2)
Building wheels for collected packages: keras-contrib
  Building wheel for keras-contrib (setup.py) … done
  Created wheel for keras-contrib: filename=keras_contrib-2.0.8-cp36-none-
any.whl size=101065
sha256=efc9f570203e265a3714b6dacd397e6114f647d6d1ba8ea78e65b47f6cdc503a
  Stored in directory: /tmp/pip-ephem-wheel-cache-9px3k7rp/wheels/11/27/c8/4ed56
de7b55f4f61244e2dc6ef3cdbaff2692527a2ce6502ba
Successfully built keras-contrib
```

[158]:
```
import numpy as np
import csv
import re
import joblib

from collections import OrderedDict
from pdfminer.high_level import extract_text

from keras.models import load_model
from keras_contrib.layers import CRF
```

```python
from keras.preprocessing.sequence import pad_sequences


max_len = 1000
model_predict = load_model('/kaggle/working/model.h5', custom_objects={"CRF": CRF, 'crf_loss': crf_layer.loss_function, 'crf_viterbi_accuracy': crf_layer.accuracy})


# Load the word2idx and tag2idx dictionaries
with open('/kaggle/working/words.pkl', 'rb') as f:
    words = joblib.load(f)

with open('/kaggle/working/tags.pkl', 'rb') as f:
    tags = joblib.load(f)

word2idx = {w: i + 1 for i, w in enumerate(words)}
tag2idx = {t: i for i, t in enumerate(tags)}
idx2tag = {i: w for w, i in tag2idx.items()}


def test_sentence_sample(test_sentence):
    results = []
    x_test_sent = pad_sequences(sequences=[[word2idx.get(w, 0) for w in tokenize(test_sentence)]],padding="post", value=0, maxlen=max_len)
    p = model_predict.predict(np.array([x_test_sent[0]]))
    p = np.argmax(p, axis=-1)
    for w, pred in zip(tokenize(test_sentence), p[0]):
        results.append([w, tags[pred]])
    return results


def tokenize(s): return s.split()



pdf_file = "/kaggle/working/inv-T191T-1630501455.pdf"
csv_file = "output.csv"

# Extract text from pdf
text = extract_text(pdf_file)
# Generate tokens
# tokens = text.split()

# print(tokens)

# check =  ' '.join(tokens)
```

```
# print(check)

results = test_sentence_sample(text)
```

```
/opt/conda/lib/python3.6/site-packages/keras_contrib/layers/crf.py:346:
UserWarning: CRF.loss_function is deprecated and it might be removed in the
future. Please use losses.crf_loss instead.
  warnings.warn('CRF.loss_function is deprecated '
/opt/conda/lib/python3.6/site-packages/keras_contrib/layers/crf.py:353:
UserWarning: CRF.accuracy is deprecated and it might be removed in the future.
Please use metrics.crf_accuracy
  warnings.warn('CRF.accuracy is deprecated and it '
/opt/conda/lib/python3.6/site-
packages/tensorflow_core/python/framework/indexed_slices.py:433: UserWarning:
Converting sparse IndexedSlices to a dense Tensor of unknown shape. This may
consume a large amount of memory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
```

[159]:
```
# results
```

[160]:
```
def process_text_file(input_text, output_file_path):
    # Split the input text into lines
    lines = input_text.split('\n')

    # Create a CSV writer
    with open(output_file_path, 'w', newline='') as output_file:
        writer = csv.writer(output_file)

        # Write the header row
        writer.writerow(["PRODUCT-NUMBER", "PRODUCT-VAT", "TOTAL-AMOUNT",␣
↪"CVR"])

        # Initialize variables
        product_number = ''
        product_vat = ''
        total_amount = ''
        cvrs = []

        # Loop through the lines in the input file
        for line in lines:
            # Strip any leading/trailing whitespace
            line = line.strip()

            # If the line contains one of the desired keys
            if "B-TOTAL-AMOUNT" in line:
                pattern = r'"(.*?)"'  # Pattern to match any characters inside␣
↪double quotes
```

```
                match = re.search(pattern, line)

                if match:
                    total_amount = match.group(1)

            elif "B-PRODUCT-VAT" in line:
                pattern = r'"(.*?)"'  # Pattern to match any characters inside␣
↪double quotes
                match = re.search(pattern, line)

                if match:
                    product_vat = match.group(1)

            elif "B-CVR" in line or "I-CVR" in line:
                # Split the line by commas
                values = line.split(',')

                # Get the second word of the line
                second_word = values[1]
                cvrs.append(second_word)

            elif "B-PRODUCT-NUMBER" in line:
                # Split the line by commas
                values = line.split(',')

                # Get the second word of the line
                product_number = values[1]

        # Write the values to the CSV file
        cvrsu = list(OrderedDict.fromkeys(cvrs))
        writer.writerow([product_number, product_vat, total_amount, ' '.
↪join(cvrsu)])

        # Return output as a dictionary
        return {"PRODUCT-NUMBER": product_number, "PRODUCT-VAT": product_vat,␣
↪"TOTAL-AMOUNT": total_amount, "CVR": cvrsu}
```

[161]:
```
def process_text_file(input_text, output_file_path):
    # Split the input text into lines
#     lines = input_text.split('\n')

    # Create a CSV writer
    with open(output_file_path, 'w', newline='') as output_file:
        writer = csv.writer(output_file)

        # Write the header row
```

```
        writer.writerow(["PRODUCT-NUMBER", "PRODUCT-VAT", "TOTAL-AMOUNT",␣
↪"CVR"])

        # Initialize variables
        product_number = []
        product_vat = []
        total_amount = []
        cvrs = []

        # Loop through the lines in the input file
        for line in input_text:
            # Strip any leading/trailing whitespace

            # If the line contains one of the desired keys
            if "B-TOTAL-AMOUNT" in line[1]:
                total_amount = line[0]

            elif "B-PRODUCT-VAT" in line[1] or "I-PRODUCT-VAT" in line[1]:
                product_vat.append(line[0])

            elif "B-CVR" in line[1] or "I-CVR" in line[1]:
                # Split the line by commas
#                 values = line.split(',')
                # Get the second word of the line
#                 second_word = values[1]
                cvrs.append(line[0])

            elif "B-PRODUCT-NUMBER" in line[1] or "I-PRODUCT-NUMBER" in line[1]:
                # Split the line by commas
#                 values = line.split(',')

                # Get the second word of the line
                product_number.append(line[0])

        # Write the values to the CSV file
        cvrsu = list(OrderedDict.fromkeys(cvrs))
        writer.writerow([product_number, product_vat, total_amount, ''.
↪join(cvrsu)])

        # Return output as a dictionary
        return {"PRODUCT-NUMBER": product_number, "PRODUCT-VAT": product_vat,␣
↪"TOTAL-AMOUNT": total_amount, "CVR": ''.join(cvrsu)}
```

[162]:
```
results
```

[162]:
```
[['Totalleverandør', 'I-BESKRIVELSE'],
 ['af', 'O'],
```

```
['fødevarer', 'B-BESKRIVELSE'],
['og', 'I-BESKRIVELSE'],
['nonfood-artikler', 'B-BESKRIVELSE'],
['House', 'I-BESKRIVELSE'],
['of', 'I-BESKRIVELSE'],
['Shawarma', 'I-BESKRIVELSE'],
['Fantastic', 'O'],
['Food', 'O'],
['Aps', 'O'],
['Rosengårdcentret', 'O'],
['Ørbækvej', 'O'],
['75', 'O'],
['5220', 'O'],
['Odense', 'O'],
['SØ', 'O'],
['Kundenr.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['79874', 'O'],
['Økologikontrolmyndighed:', 'O'],
['DK-ØKO-100', 'O'],
['Kopi', 'O'],
['*10-3162056*', 'O'],
['Faktura', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['Bogføringsdato', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
```

```
['.', 'O'],
['.', 'O'],
['Rute', 'O'],
['Sælger', 'O'],
['Betalingsbetingels', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['Betales', 'O'],
['inden', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['Side', 'O'],
['10-3162056', 'O'],
['09-07-21', 'O'],
['55', 'O'],
['CGOPHA/CGOCLG', 'O'],
['30', 'O'],
['Dage', 'O'],
['Netto', 'O'],
['08-08-21', 'O'],
['1', 'O'],
['59007', 'O'],
['Nummer', 'O'],
['Beskrivelse', 'O'],
['Mærke', 'O'],
['Antal', 'O'],
['Enhed', 'O'],
['Salgspris', 'O'],
['Beløb', 'O'],
['206873', 'O'],
['Agurk', 'B-BESKRIVELSE'],
['m/Film', 'I-BESKRIVELSE'],
['(NL)', 'I-BESKRIVELSE'],
['206928', 'O'],
```

```
['Koriander', 'B-BESKRIVELSE'],
['i', 'I-BESKRIVELSE'],
['Bundt', 'I-BESKRIVELSE'],
['(KE)', 'O'],
['206863', 'O'],
['Løg', 'B-BESKRIVELSE'],
['i', 'I-BESKRIVELSE'],
['PS', 'O'],
['(DK)', 'O'],
['229956', 'O'],
['Persille', 'B-BESKRIVELSE'],
['Bredbladet', 'I-BESKRIVELSE'],
['i', 'I-BESKRIVELSE'],
['Bundt', 'I-BESKRIVELSE'],
['(IT)', 'O'],
['206826', 'O'],
['Salat', 'B-BESKRIVELSE'],
['Iceberg', 'I-BESKRIVELSE'],
['(NL)', 'I-BESKRIVELSE'],
['218441', 'O'],
['Tomat', 'B-BESKRIVELSE'],
['Udenlandsk', 'I-BESKRIVELSE'],
['57-67', 'I-BESKRIVELSE'],
['(NL)', 'O'],
['67418', 'O'],
['Yoghurt', 'B-BESKRIVELSE'],
['Græsk', 'I-BESKRIVELSE'],
['Inspireret', 'I-BESKRIVELSE'],
['10%', 'I-BESKRIVELSE'],
['3232', 'O'],
['Yoghurt', 'B-BESKRIVELSE'],
['Sødmælk', 'I-BESKRIVELSE'],
['Naturel', 'I-BESKRIVELSE'],
['3,5%', 'I-BESKRIVELSE'],
['2435', 'I-BESKRIVELSE'],
['Salatmayonnaise', 'B-BESKRIVELSE'],
['50%', 'I-BESKRIVELSE'],
['42893', 'I-BESKRIVELSE'],
['Salatmayonnaise', 'B-BESKRIVELSE'],
['Portion', 'I-BESKRIVELSE'],
['42891', 'I-BESKRIVELSE'],
['Tomatketchup', 'B-BESKRIVELSE'],
['Portionspakning', 'I-BESKRIVELSE'],
['72647', 'O'],
['Sødmælk', 'B-BESKRIVELSE'],
['UHT', 'I-BESKRIVELSE'],
['3,5%', 'I-BESKRIVELSE'],
```

['316302', 'O'],
['Fritureolie', 'B-BESKRIVELSE'],
['316304', 'I-BESKRIVELSE'],
['Rapsolie', 'B-BESKRIVELSE'],
['21390', 'I-BESKRIVELSE'],
['Eddike', 'B-BESKRIVELSE'],
['u/Konserveringsmid', 'I-BESKRIVELSE'],
['5%', 'I-BESKRIVELSE'],
['Kølevarer', 'O'],
['GRØNT', 'O'],
['KRYDDER', 'O'],
['HØJVANG', 'O'],
['KRYDDER', 'O'],
['GRØNT', 'O'],
['GRØNT', 'O'],
['SOL', 'I-BESKRIVELSE'],
['ARLA', 'O'],
['PRO', 'O'],
['KRAFT', 'O'],
['BÄHNCKE', 'O'],
['BÄHNCKE', 'O'],
['GOURMET', 'O'],
['DELTA', 'O'],
['DELTA', 'O'],
['LAGERBE', 'O'],
['Tørvarer', 'O'],
['Frostvarer', 'O'],
['3', 'O'],
['2', 'O'],
['2', 'O'],
['2', 'O'],
['8', 'O'],
['4', 'O'],
['1', 'O'],
['10', 'O'],
['1', 'O'],
['3', 'O'],
['3', 'O'],
['12', 'O'],
['6', 'O'],
['2', 'O'],
['4', 'O'],
['STK', 'O'],
['PS', 'O'],
['PS', 'O'],
['BT', 'O'],
['KRT', 'O'],

['KRT', 'O'],
['SP', 'O'],
['(5', 'O'],
['KG)', 'O'],
['SP', 'O'],
['(5', 'O'],
['LTR)', 'O'],
['SP', 'O'],
['(10', 'O'],
['KG)', 'O'],
['ÆSK', 'O'],
['(126x25', 'O'],
['G)', 'O'],
['ÆSK', 'O'],
['(126x25', 'O'],
['G)', 'O'],
['STK', 'O'],
['(1', 'O'],
['LTR)', 'O'],
['DK', 'O'],
['(10', 'O'],
['LTR)', 'O'],
['DK', 'O'],
['(10', 'O'],
['LTR)', 'O'],
['DK', 'O'],
['(10', 'O'],
['LTR)', 'O'],
['4,74', 'O'],
['11,96', 'O'],
['6,35', 'O'],
['10,31', 'O'],
['65,40', 'O'],
['67,62', 'O'],
['83,35', 'O'],
['60,45', 'O'],
['149,00', 'O'],
['125,50', 'O'],
['129,05', 'O'],
['7,75', 'O'],
['108,00', 'O'],
['108,00', 'O'],
['39,50', 'O'],
['14,22', 'O'],
['23,92', 'O'],
['12,70', 'O'],
['20,62', 'O'],

['523,20', 'O'],
['270,48', 'O'],
['83,35', 'O'],
['604,50', 'O'],
['149,00', 'O'],
['376,50', 'O'],
['387,15', 'O'],
['93,00', 'O'],
['648,00', 'O'],
['216,00', 'O'],
['158,00', 'O'],
['*', 'O'],
['*', 'O'],
['*', 'O'],
['*', 'O'],
['*', 'O'],
['*', 'O'],
['*', 'O'],
['*', 'O'],
['*', 'O'],
['*', 'O'],
['*', 'O'],
['*', 'O'],
['*', 'O'],
['*', 'O'],
['*', 'O'],
['*', 'O'],
['234734', 'O'],
['PommesFrites', 'B-BESKRIVELSE'],
['7x7', 'I-BESKRIVELSE'],
['mm', 'I-BESKRIVELSE'],
['FARMFRIT', 'O'],
['12', 'O'],
['KRT', 'O'],
['(5x2,5', 'O'],
['KG)', 'O'],
['106,25', 'O'],
['1.275,00', 'O'],
['Moms', 'O'],
['nr.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],
['.', 'O'],

```
['.', 'O'],
['.', 'O'],
['39902060', 'O'],
['+71<', 'O'],
['000010316205607', 'O'],
['+89136296<', 'O'],
['I', 'O'],
['alt', 'O'],
['DKK', 'O'],
['25%', 'O'],
['moms', 'O'],
['I', 'O'],
['alt', 'O'],
['DKK', 'O'],
['inkl.', 'O'],
['moms', 'O'],
['4.855,64', 'O'],
['1.213,91', 'O'],
['6.069,55', 'O'],
['BC', 'O'],
['Catering', 'O'],
['Grossisten', 'O'],
['A/S', 'O'],
['Blækhatten', 'O'],
['10', 'O'],
['5220', 'O'],
['Odense', 'O'],
['SØ', 'O'],
['Tlf.', 'O'],
['63', 'O'],
['15', 'O'],
['88', 'O'],
['55', 'O'],
['CVR', 'O'],
['10', 'O'],
['83', 'B-BESKRIVELSE'],
['70', 'I-BESKRIVELSE'],
['49', 'I-BESKRIVELSE'],
['www.bccatering.dk', 'I-BESKRIVELSE']]
```

[163]:
```
process_text_file(results, 'input.csv')
```

[163]:
```
{'PRODUCT-NUMBER': [], 'PRODUCT-VAT': [], 'TOTAL-AMOUNT': [], 'CVR': ''}
```

[164]:
```
# !cd /kaggle/working
```

[165]:
```
# !cd ..
```

[166]:
```
# from IPython.display import FileLink
# FileLink(r'model.h5')
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[167]:
```
+
```

```
  File "<ipython-input-167-0b024bbfe84e>", line 1
    +
     ^
SyntaxError: invalid syntax
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: