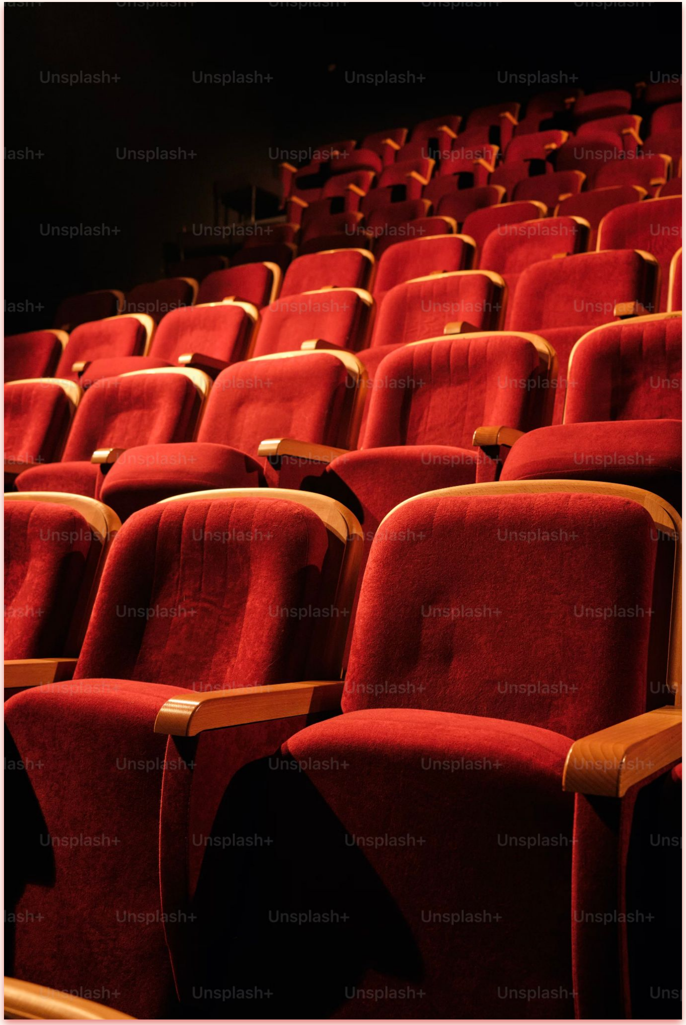
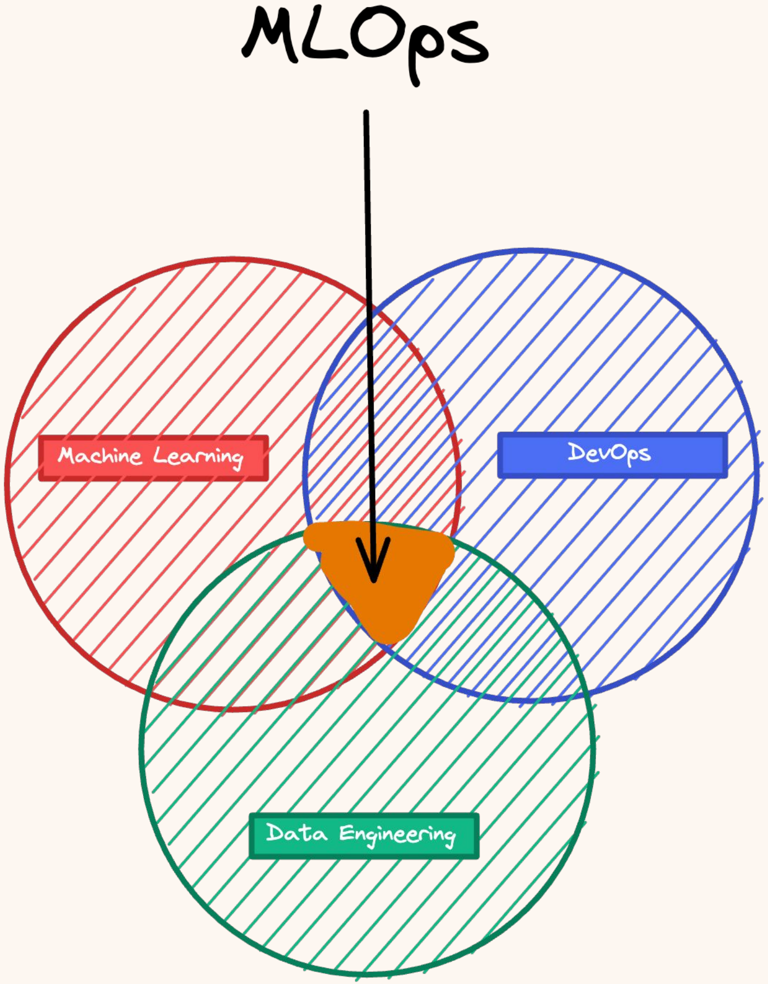
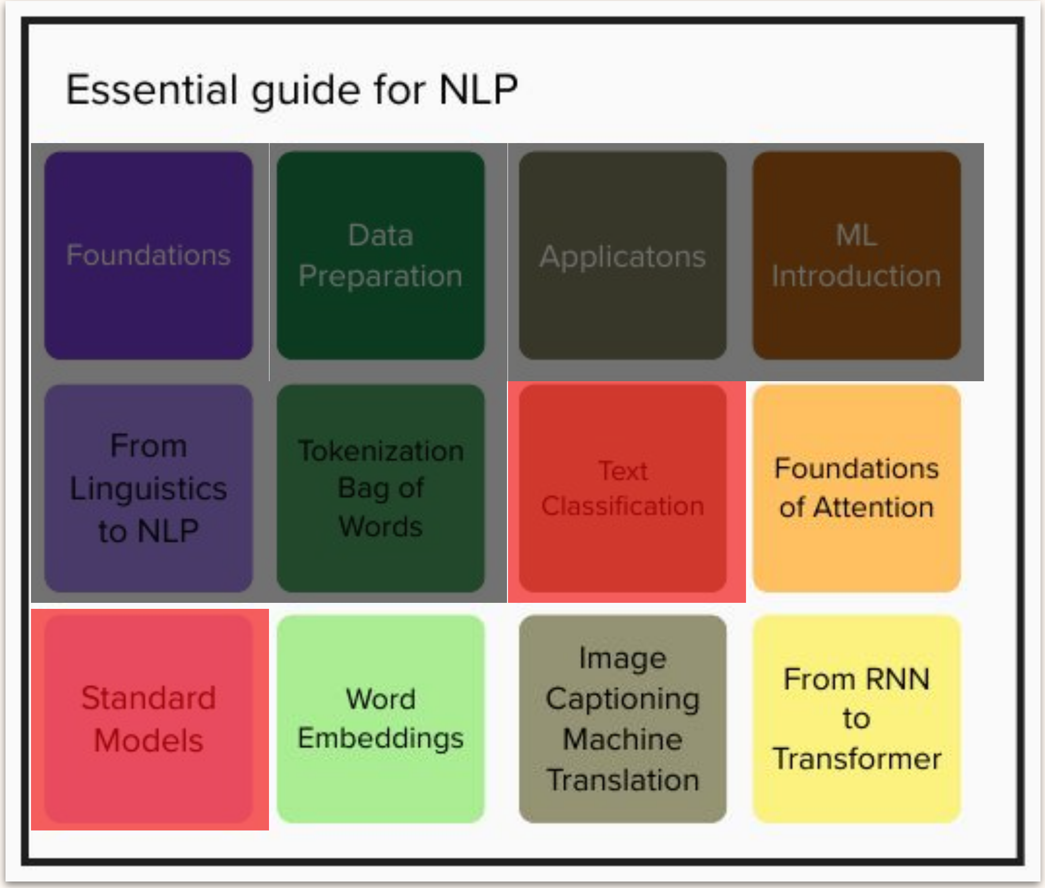
Essential Guide for NLP

Construct a Neural Bag-of-Words Framework for Evaluating Sentiments



DCA0305
ivanovitch.silva@ufrn.br

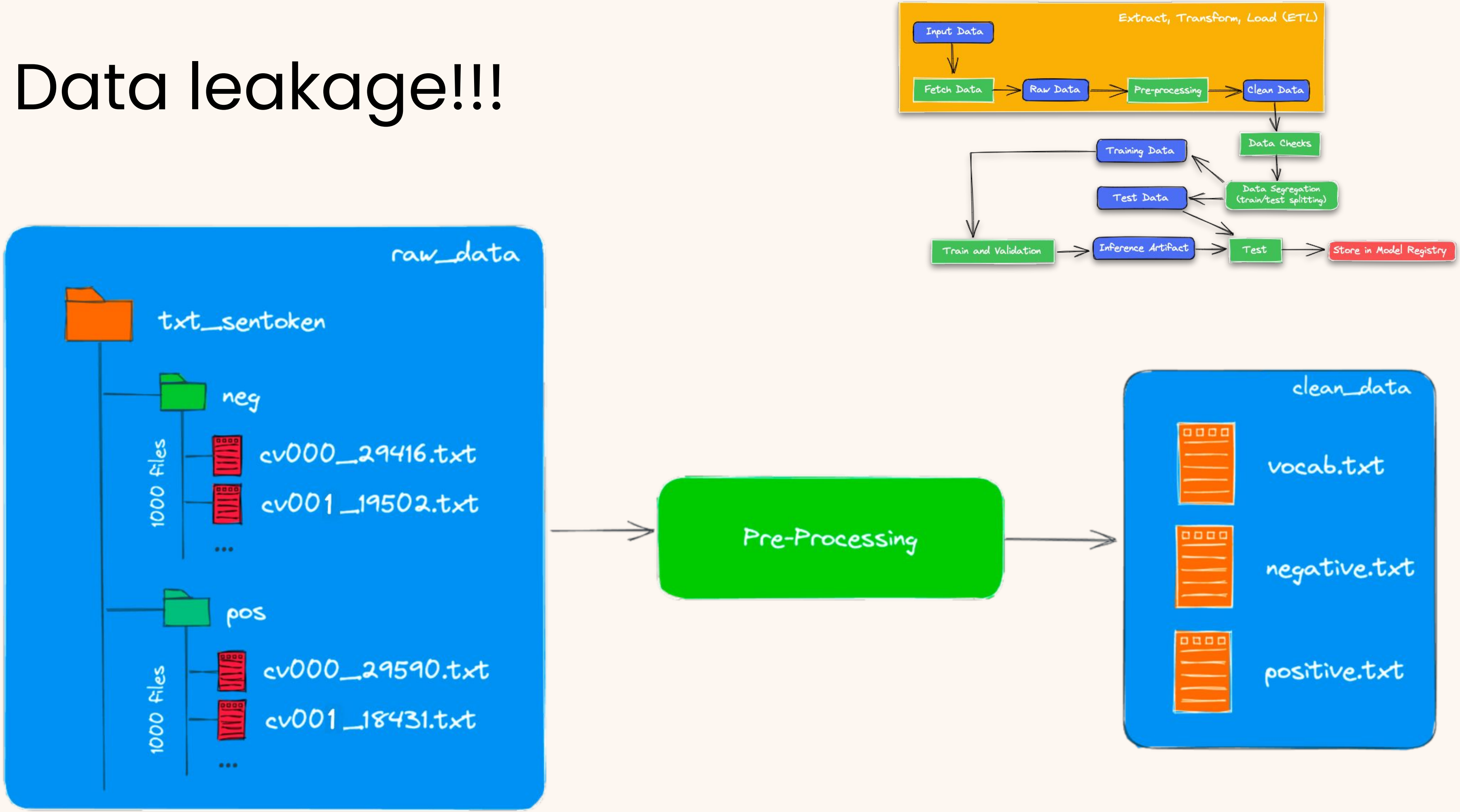


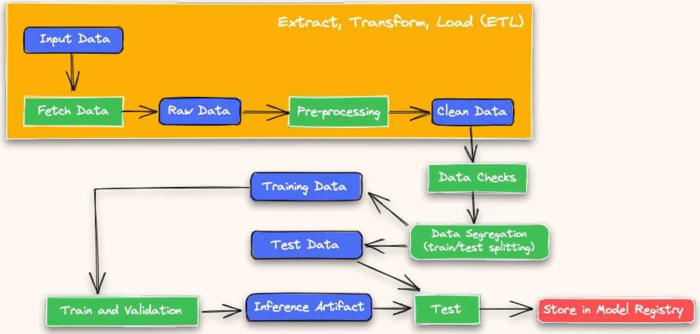


Essential guide for NLP

Foundations	Data Preparation	Applicatons	ML Introduction
From Linguistics to NLP	Tokenization Bag of Words	Text Classification	Foundations of Attention
Standard Models	Word Embeddings	Image Captioning Machine Translation	From RNN to Transformer

Data leakage!!!





Extract, Transform, Load (ETL)

Input Data

Fetch Data

Raw Data

Pre-processing

Clean Data

Data Checks

Data Segregation (train/test splitting)

Training Data

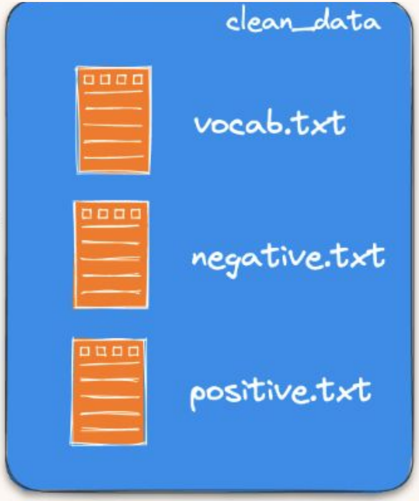
Test Data

Train and Validation

Inference Artifact

Test

Store in Model Registry

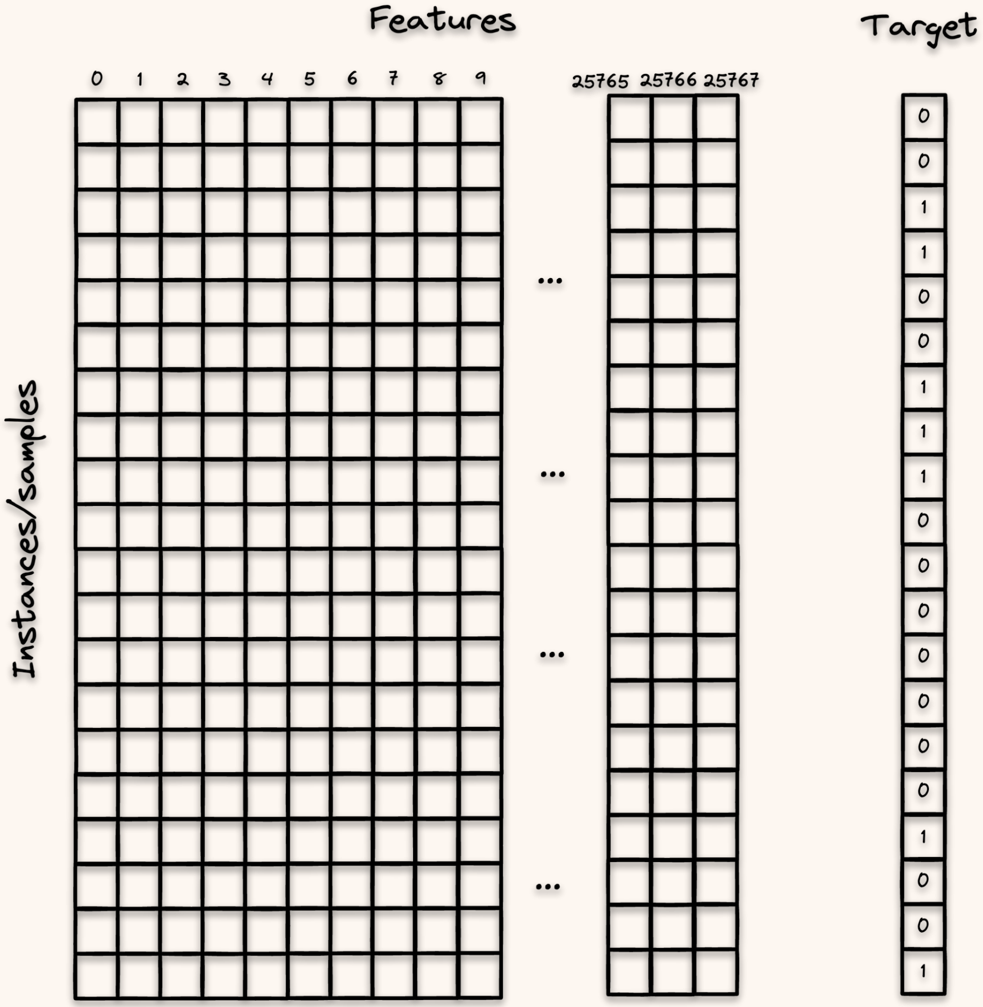


clean_data

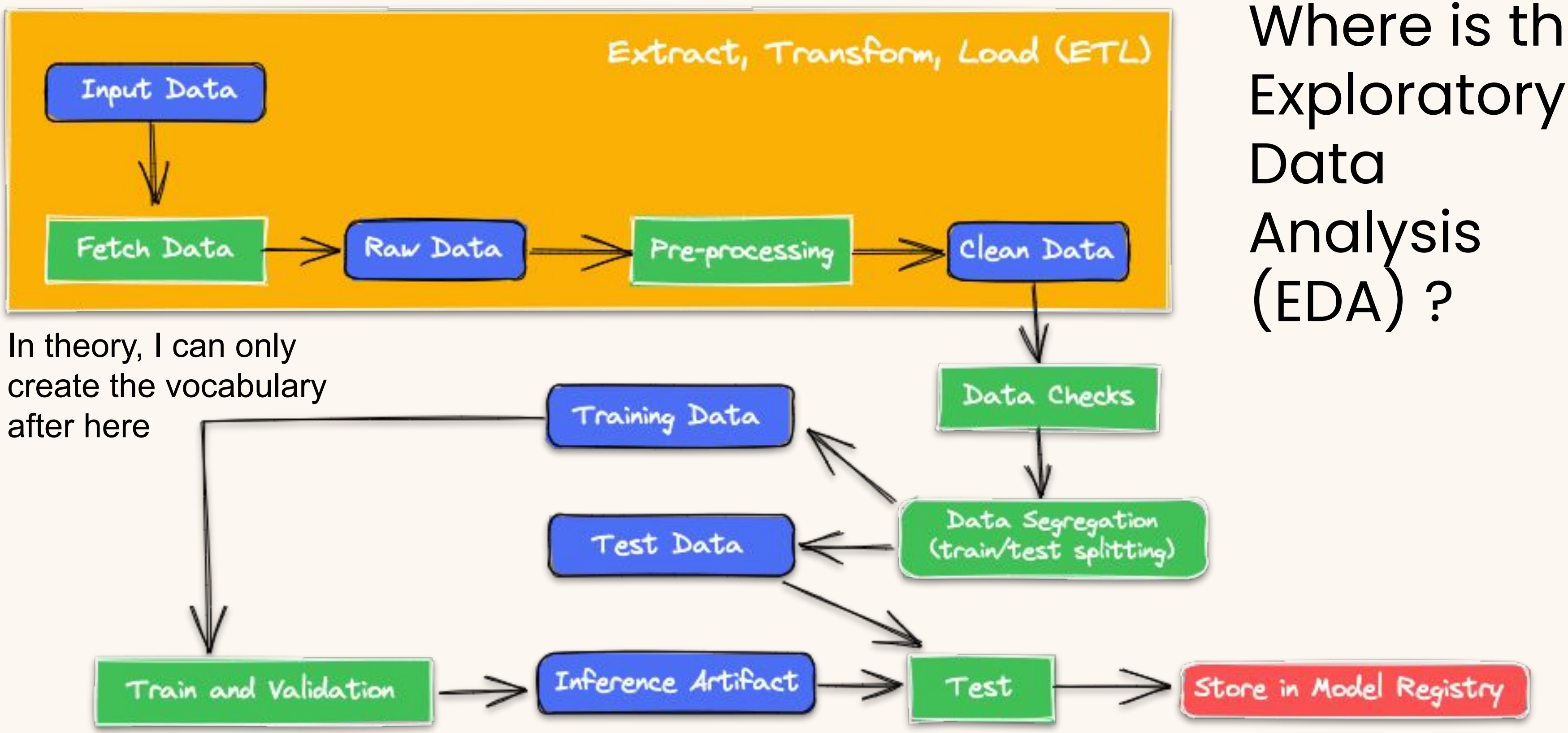
vocab.txt

negative.txt

positive.txt

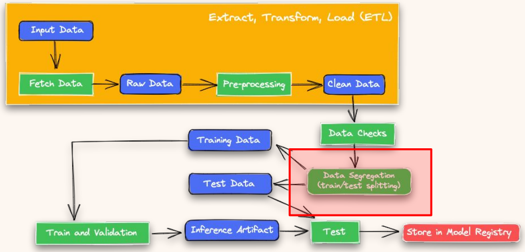


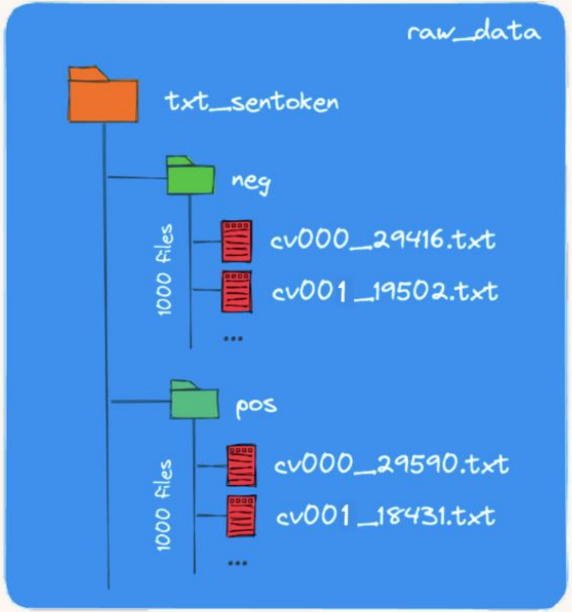
Where is the Exploratory Data Analysis (EDA) ?



In theory, I can only create the vocabulary after here

Vocabulary Creation only using Train Data





```
# load all docs in a directory
def process_docs(directory, vocab):
# walk through all files in the folder
    for filename in listdir(directory):
        # skip any reviews in the test set
        if filename.startswith('cv9'):
            continue
        # create the full path of the file to open
        path = directory + '/' + filename
        # add doc to vocab
        add_doc_to_vocab(path, vocab)
```

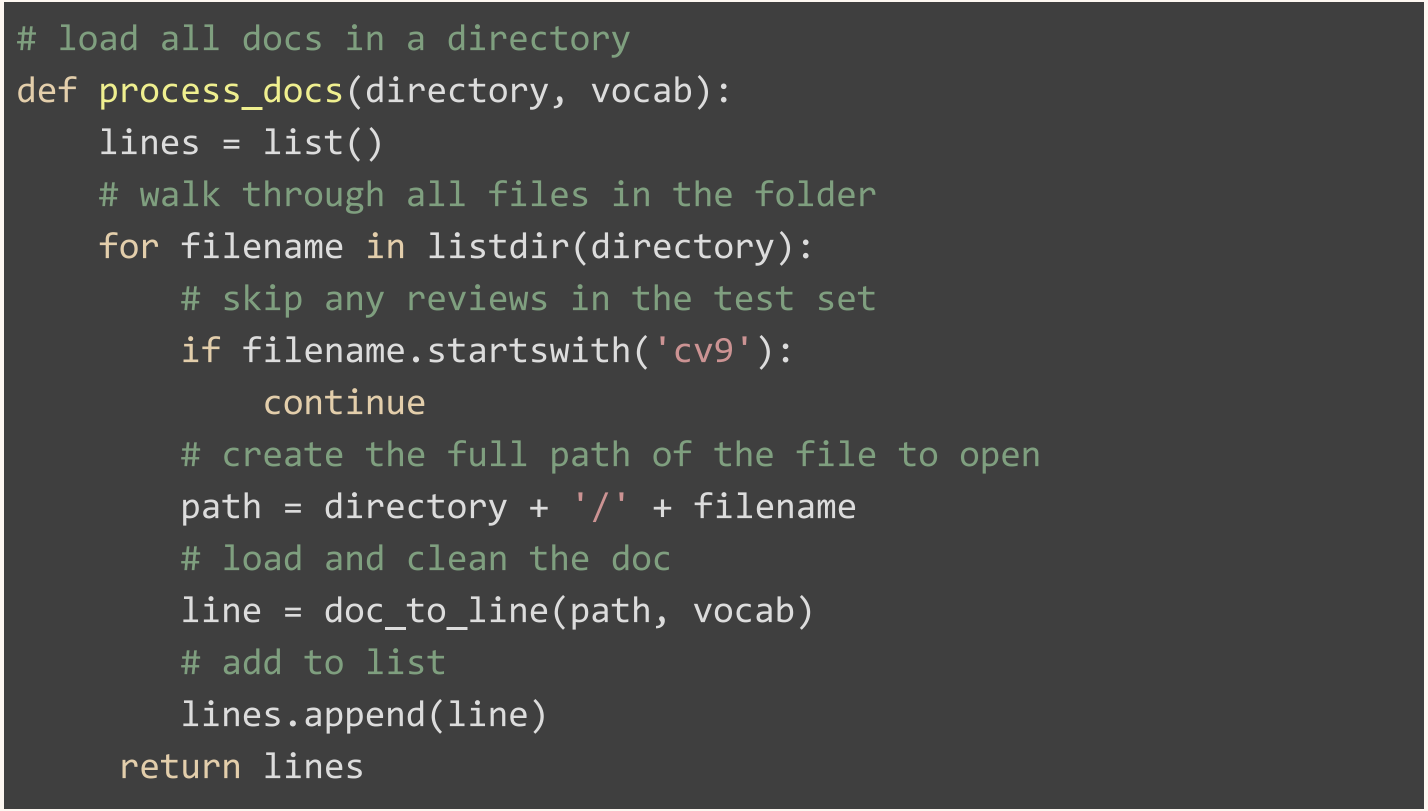
Converting Reviews into Lines of Tokens

```
# load the vocabulary
vocab_filename = 'vocab.txt'
vocab = load_doc(vocab_filename)
vocab = vocab.split()
vocab = set(vocab)
# load all training reviews
docs, labels = load_clean_dataset(vocab)
# summarize what we have
print(len(docs), len(labels))
```

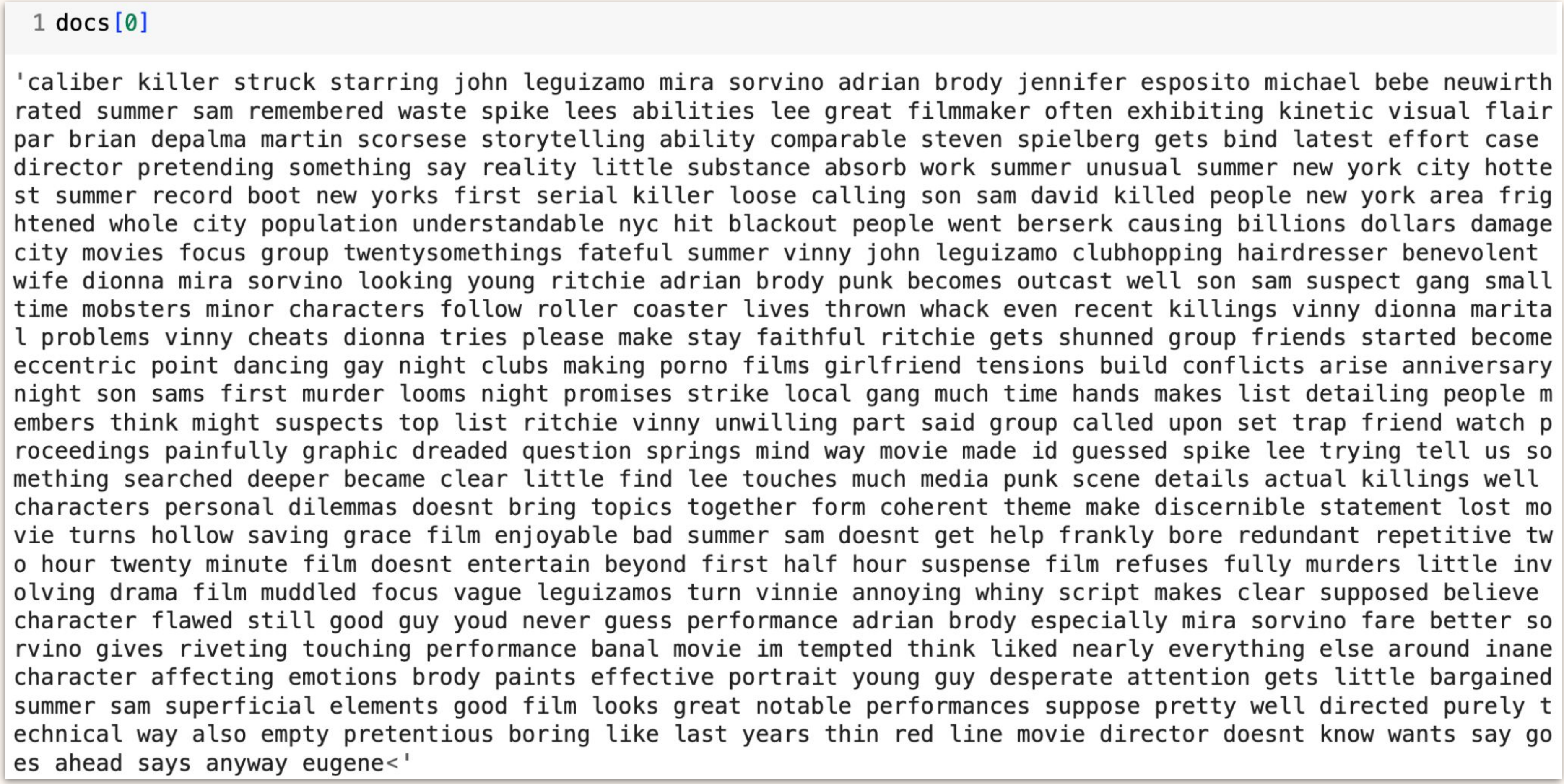
1800,1800

```
# load and clean a dataset
def load_clean_dataset(vocab):
    # load documents
    neg = process_docs('txt_sentoken/neg', vocab)
    pos = process_docs('txt_sentoken/pos', vocab)
    docs = neg + pos
    # prepare labels
    labels = [0 for _ in range(len(neg))] + [1 for _
in range(len(pos))]
    return docs, labels
```

Converting Reviews into Lines of Tokens



```python
# load all docs in a directory
def process_docs(directory, vocab):
    lines = list()
    # walk through all files in the folder
    for filename in listdir(directory):
        # skip any reviews in the test set
        if filename.startswith('cv9'):
            continue
        # create the full path of the file to open
        path = directory + '/' + filename
        # load and clean the doc
        line = doc_to_line(path, vocab)
        # add to list
        lines.append(line)
     return lines
```

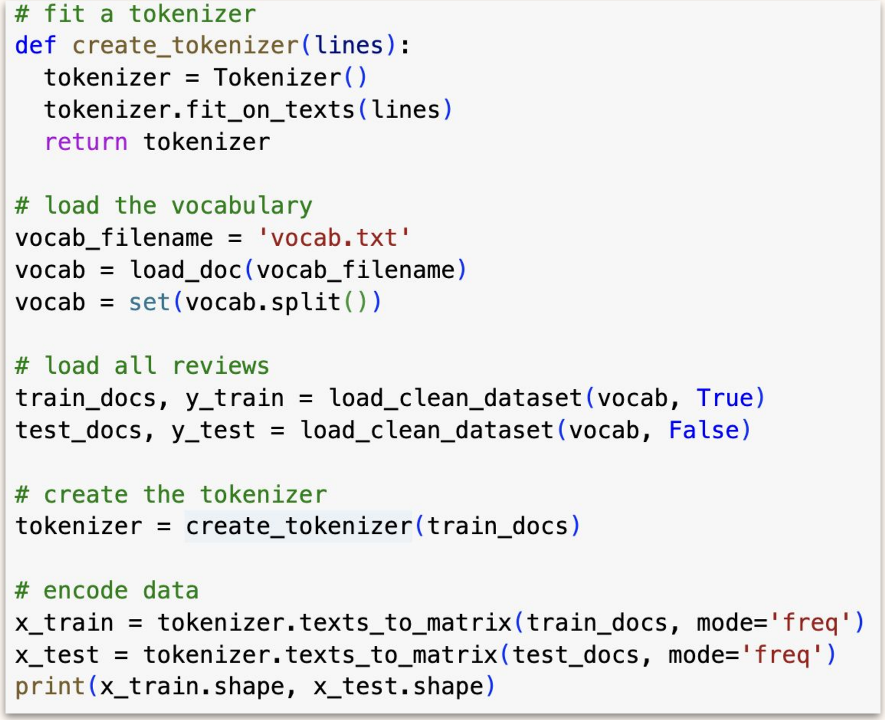


```python
docs[0]
```

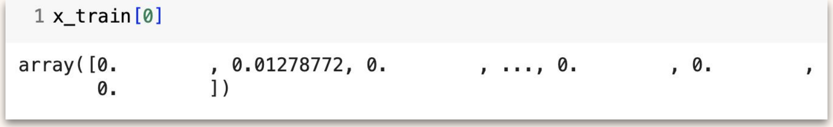
```
'caliber killer struck starring john leguizamo mira sorvino adrian brody jennifer esposito michael bebe neuwirth
rated summer sam remembered waste spike lees abilities lee great filmmaker often exhibiting kinetic visual flair
par brian depalma martin scorsese storytelling ability comparable steven spielberg gets bind latest effort case
director pretending something say reality little substance absorb work summer unusual summer new york city hotte
st summer record boot new yorks first serial killer loose calling son sam david killed people new york area frig
htened whole city population understandable nyc hit blackout people went berserk causing billions dollars damage
city movies focus group twentysomethings fateful summer vinny john leguizamo clubhopping hairdresser benevolent
wife dionna mira sorvino looking young ritchie adrian brody punk becomes outcast well son sam suspect gang small
time mobsters minor characters follow roller coaster lives thrown whack even recent killings vinny dionna marita
l problems vinny cheats dionna tries please make stay faithful ritchie gets shunned group friends started become
eccentric point dancing gay night clubs making porno films girlfriend tensions build conflicts arise anniversary
night son sams first murder looms night promises strike local gang much time hands makes list detailing people m
embers think might suspects top list ritchie vinny unwilling part said group called upon set trap friend watch p
roceedings painfully graphic dreaded question springs mind way movie made id guessed spike lee trying tell us so
mething searched deeper became clear little find lee touches much media punk scene details actual killings well
characters personal dilemmas doesnt bring topics together form coherent theme make discernible statement lost mo
vie turns hollow saving grace film enjoyable bad summer sam doesnt get help frankly bore redundant repetitive tw
o hour twenty minute film doesnt entertain beyond first half hour suspense film refuses fully murders little inv
olving drama film muddled focus vague leguizamos turn vinnie annoying whiny script makes clear supposed believe
character flawed still good guy youd never guess performance adrian brody especially mira sorvino fare better so
rvino gives riveting touching performance banal movie im tempted think liked nearly everything else around inane
character affecting emotions brody paints effective portrait young guy desperate attention gets little bargained
summer sam superficial elements good film looks great notable performances suppose pretty well directed purely t
echnical way also empty pretentious boring like last years thin red line movie director doesnt know wants say go
es ahead says anyway eugene<'
```

Transforming Movie Reviews into Bag-of-Words Vectors

tip: the previous step is repeated here



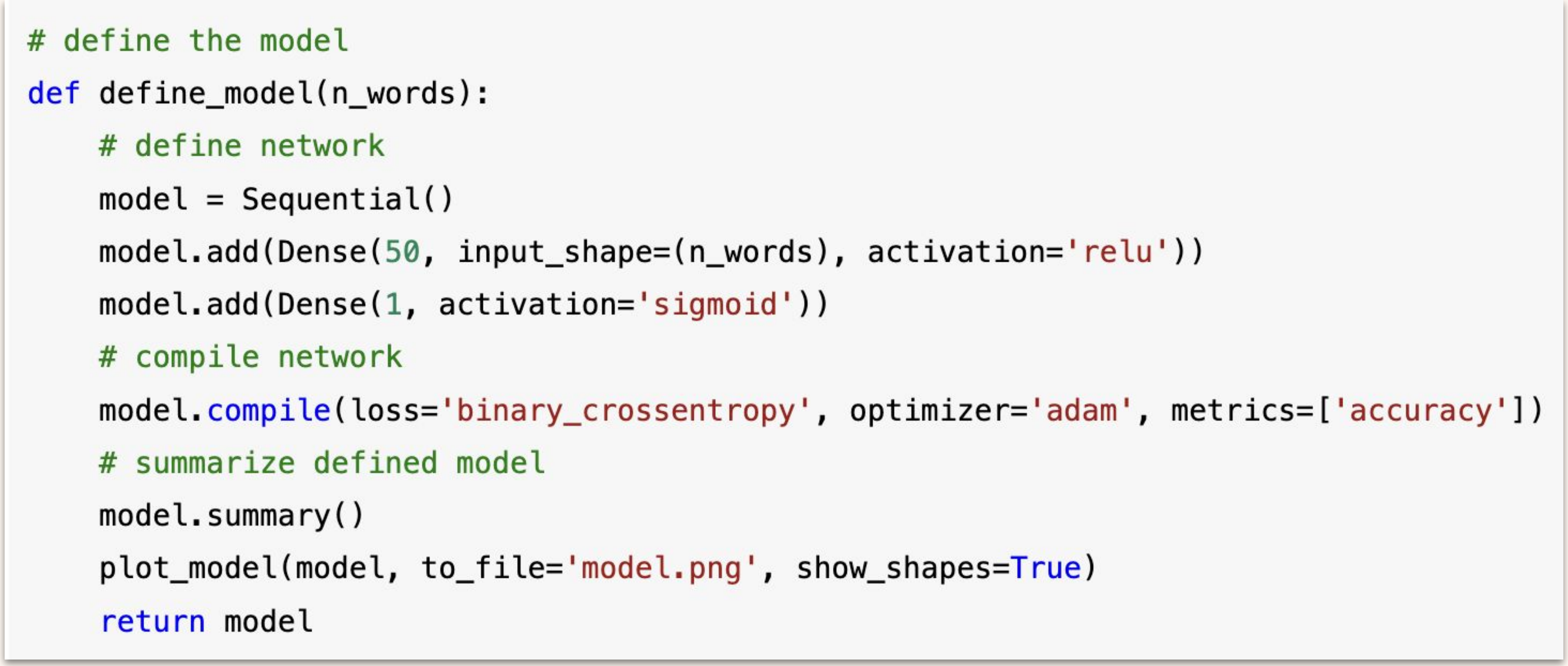
```python
# fit a tokenizer
def create_tokenizer(lines):
  tokenizer = Tokenizer()
  tokenizer.fit_on_texts(lines)
  return tokenizer

# load the vocabulary
vocab_filename = 'vocab.txt'
vocab = load_doc(vocab_filename)
vocab = set(vocab.split())

# load all reviews
train_docs, y_train = load_clean_dataset(vocab, True)
test_docs, y_test = load_clean_dataset(vocab, False)

# create the tokenizer
tokenizer = create_tokenizer(train_docs)

# encode data
x_train = tokenizer.texts_to_matrix(train_docs, mode='freq')
x_test = tokenizer.texts_to_matrix(test_docs, mode='freq')
print(x_train.shape, x_test.shape)
```



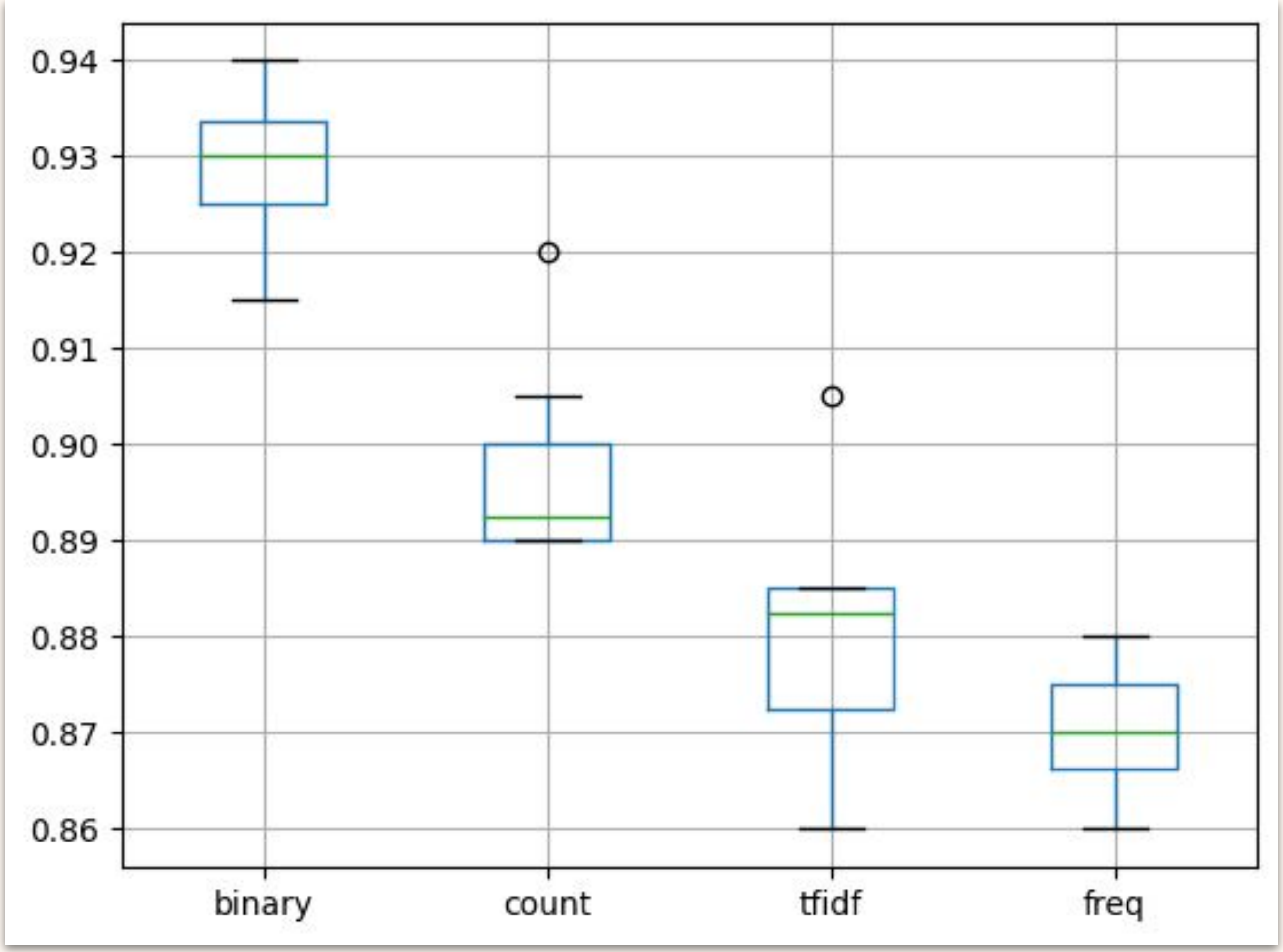
```python
x_train[0]
```

```
array([0.        , 0.01278772, 0.        , ..., 0.        , 0.        ,
       0.        ])
```

Sentiment Analysis Models



```python
# define the model
def define_model(n_words):
    # define network
    model = Sequential()
    model.add(Dense(50, input_shape=(n_words), activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
    # compile network
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    # summarize defined model
    model.summary()
    plot_model(model, to_file='model.png', show_shapes=True)
    return model
```


0.94
0.93
0.92
0.91
0.90
0.89
0.88
0.87
0.86
binary
count
tfidf
freq