```
import pandas as pd
import gensim
import numpy as np
from gensim.corpora import Dictionary
import keras
import json
```

```
Using TensorFlow backend.
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x
via the `%tensorflow_version 1.x` magic: more info.

```
def texts_to_indices(text, dictionary):
    """
    Given a list of tokens (text) and a gensim dictionary, return a list
    of token ids.
    """
    result = list(map(lambda x: token_to_index(x, dictionary), text))
    return list(filter(None, result))

def token_to_index(token, dictionary):
    """
    Given a token and a gensim dictionary, return the token index
    if in the dictionary, None otherwise.
    Reserve index 0 for padding.
    """
    if token not in dictionary.token2id:
        return None
    return dictionary.token2id[token] + 1

def tokenize(text):
    # for each token in the text (the result of text.split(),
    # apply a function that strips punctuation and converts to lower case.
    tokens = map(lambda x: x.strip(',.&').lower(), text.split())
    # get rid of empty tokens
    tokens = list(filter(None, tokens))
    return tokens



# get dictionary
  df = pd.read_csv("yelp_reviews.csv", encoding='utf-8', engine='python', error_bad_l
  text = df['text'].values.tolist()

  def uni_and_bigrams(text):
    # our unigrams are our tokens
    unigrams=tokenize(text)
    # the bigrams just contatenate 2 adjacent tokens with _ in between
    bigrams=list(map(lambda x: '_'.join(x), zip(unigrams, unigrams[1:])))
    # returning a list containing all 1 and 2-grams
    return unigrams+bigrams
```

```
  # texts = list(map(tokenize, text))
  # mydict = gensim.corpora.Dictionary(texts)
  # mydict.save('yelp.dict')


tokenized_texts=list(map(uni_and_bigrams, text))

my_bigram_dict = gensim.corpora.Dictionary(tokenized_texts)
my_bigram_dict.save('my_bigram_dict.dict')
```

```
/usr/local/lib/python3.6/dist-packages/smart_open/smart_open_lib.py:398: UserWarn
  'See the migration notes for details: %s' % _MIGRATION_NOTES_URL
```

```
new_model = keras.models.load_model('yelp_cnn_bigram.model')
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tens
```

```
def cnn_predict(document):

  #predict label and corresponding probability
  test_predict = texts_to_indices(document, my_bigram_dict)
  len_predict = len(test_predict)
  max_len = 453
  final_predict = [0 for i in range(0,max_len-len_predict)]
  final_predict.extend(test_predict)
  input_ = np.array([final_predict])
  input_.shape
  predicted_label = new_model.predict_classes(input_)
  predicted_prob = new_model.predict(input_)

  print("Predicted_label =%s" % (predicted_label))
  print("Predicted_probability (confidence of predicted label) =%s" % (predicted_prob

  #Get results in json format and save
  with open('svm_predcition.json', 'w') as fp:
        json.dump(str(predicted_label[0]), fp)
  # print out saved dictionary
  print("")
  print("Saved as json")

  return predicted_label, predicted_prob
```

```
# tmp_fname = 'yelp.dict'
# my_dict = Dictionary.load_from_text(tmp_fname)
```

```
document = "this place is so good we went back twice in the row. The chicken was amaz
cnn_predict(document)
```

```
Predicted_label =[5]
Predicted_probability (confidence of predicted label) =[[2.1573092e-06 1.9489498e
  9.9901164e-01]]

Saved as json
(array([5]),
 array([[2.1573092e-06, 1.9489498e-04, 2.3761902e-06, 7.5632288e-06,
         7.8129920e-04, 9.9901164e-01]], dtype=float32))
```