I started working on the challenge on Saturday and the steps I undertook are:

Setup local environment for NLTK processing by installing the important packages such as NLTK, SCIPY, PANDAS, SKLEARN, NUMPY and KERAS. Normally all of the packages aren't available in a single Python software or IDE; and either they have to be individually installed into python or you get them assembled as a group in anaconda excluding Keras.

After setting up the environment, I studied and analyzed the preprocessing, evaluation, model, chabot and main codes, which posses defined functions of five, five, eight, three and two respectively, that are present within the python files.

My study of the codes is incomplete as I got stuck in a case in which a little tip from an experienced colleague would have made a big difference to proceed in finding a solution to the problem.

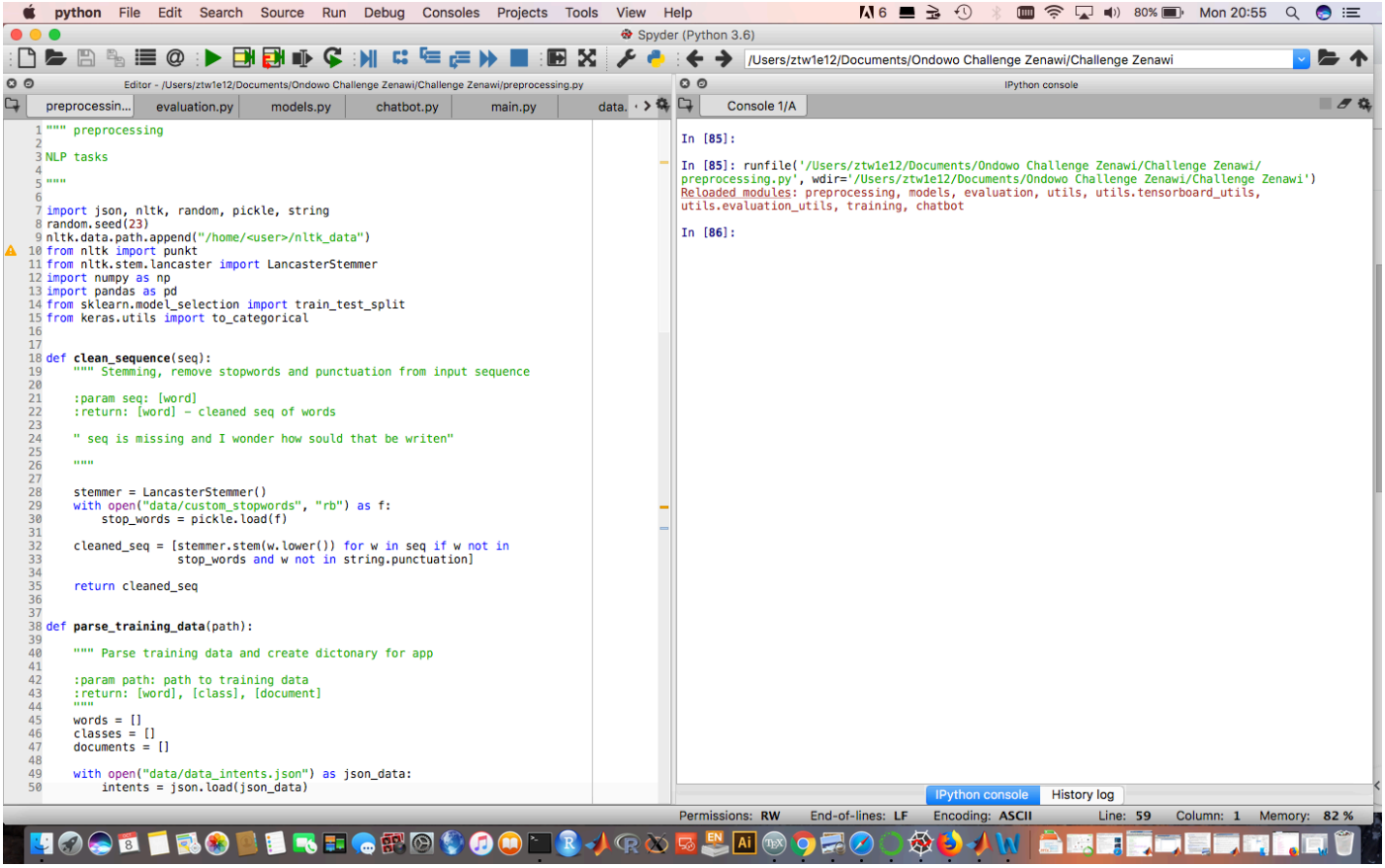
After running the preprocessing, evaluation, model and chabot scripts, I couldn't observe (notice) any bugs or error messages in the command window. But, after executing the 'main' script I found an error pointing to **LINE 59** of the preprocessing script linked to LINE 43 of the 'main' script. I would love to carry on working on the bug and resolve the challenge, but I run out of time to do it. Moreover, given some more time along with some communications for help I need for the challenge I would write the requested algorithm and test, which model works better on the supplied data. As I have indicated before, I squeezed this challenge along with the commitments I have from my current work.

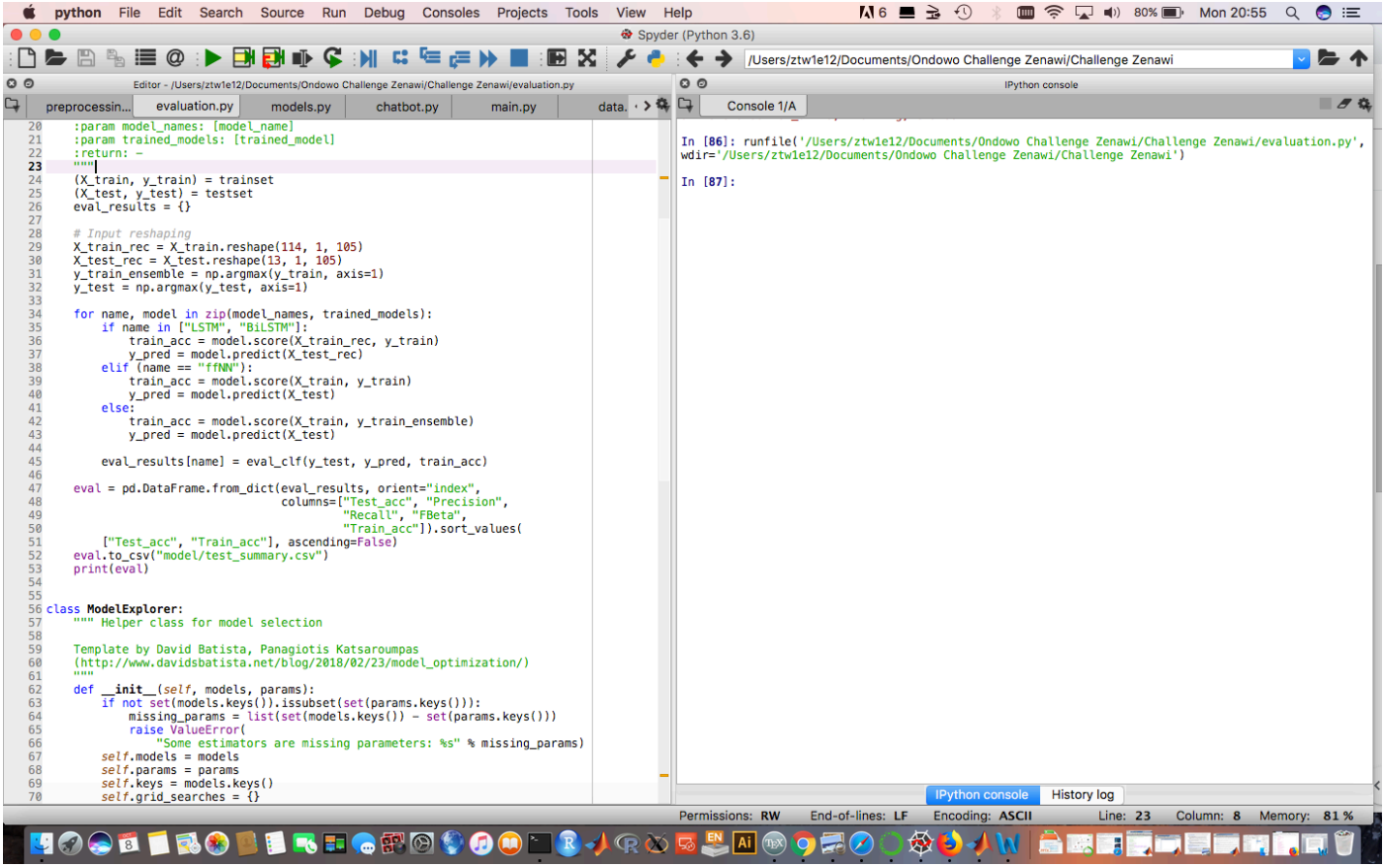
Please find enclosed below some errors and messages thrown to the command window during the computation of the challenge.

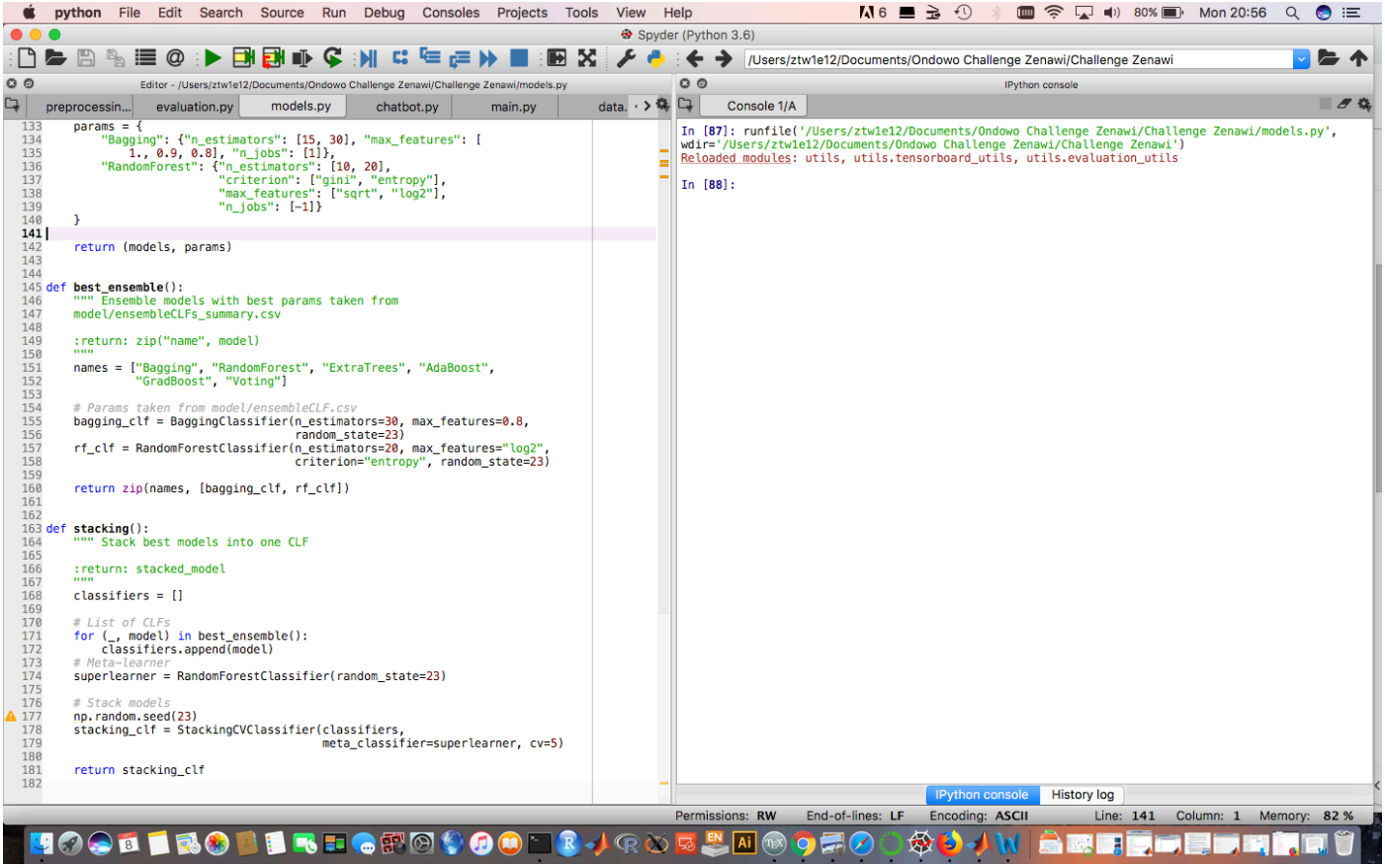
As for the question about awareness of having a little training data, here below is a statement about my understanding and response for that:

The larger the training data set is made, the better the model will be trained. Every data received in data science is normally split in to two data subsets: the training data set and the test data set. The training set is the data in which a model is trained at; the test set on the other hand is the data in which the performance of the model is measured at. Data needs to be randomized before we split it into training and test data sets. This would make sure we don't use data of different characteristics for training and testing a model.

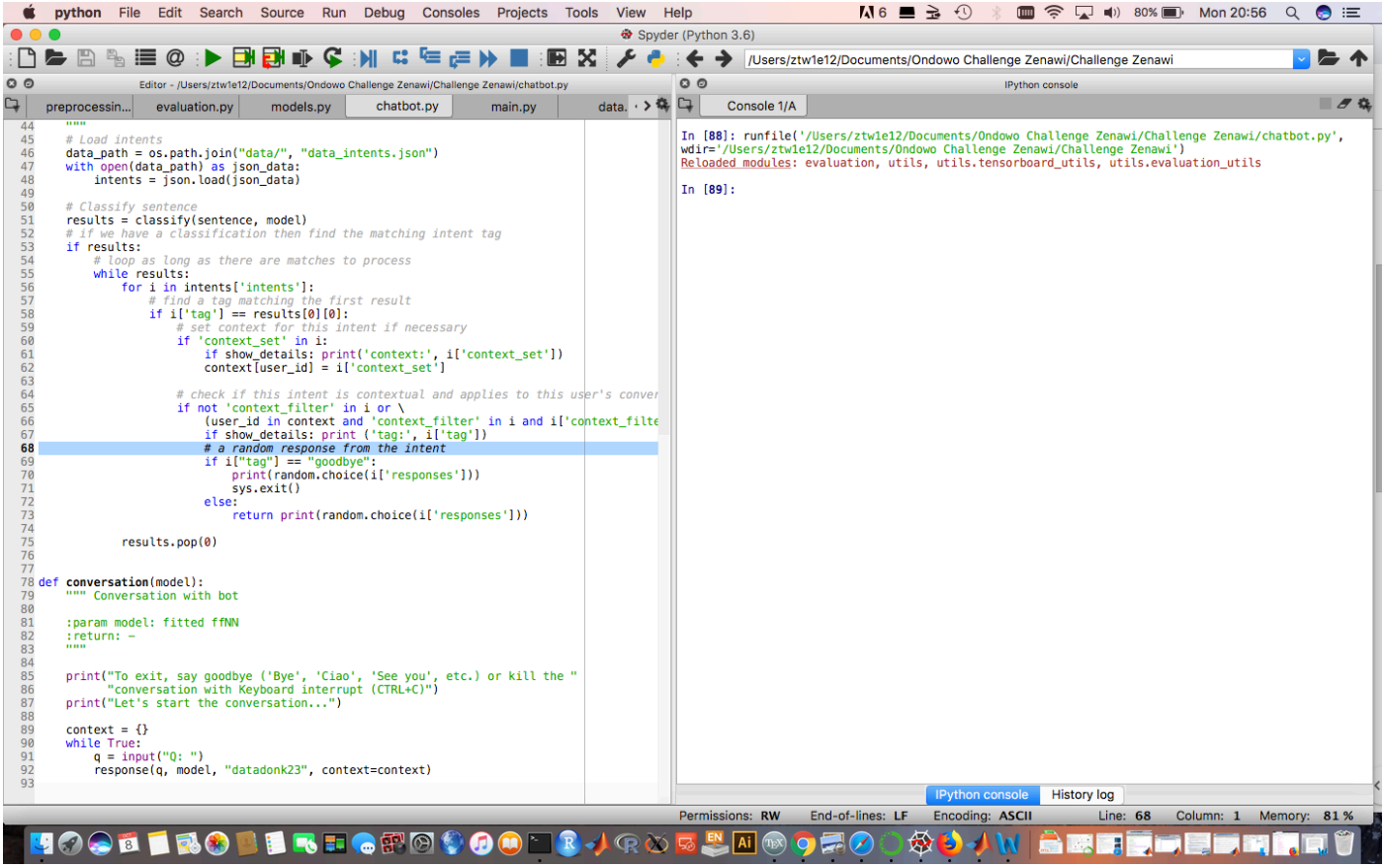
However, in the case of small data set, a more sophisticated approach like cross validation should be carried out to better train a model and have better confidence on the evaluation metrics.

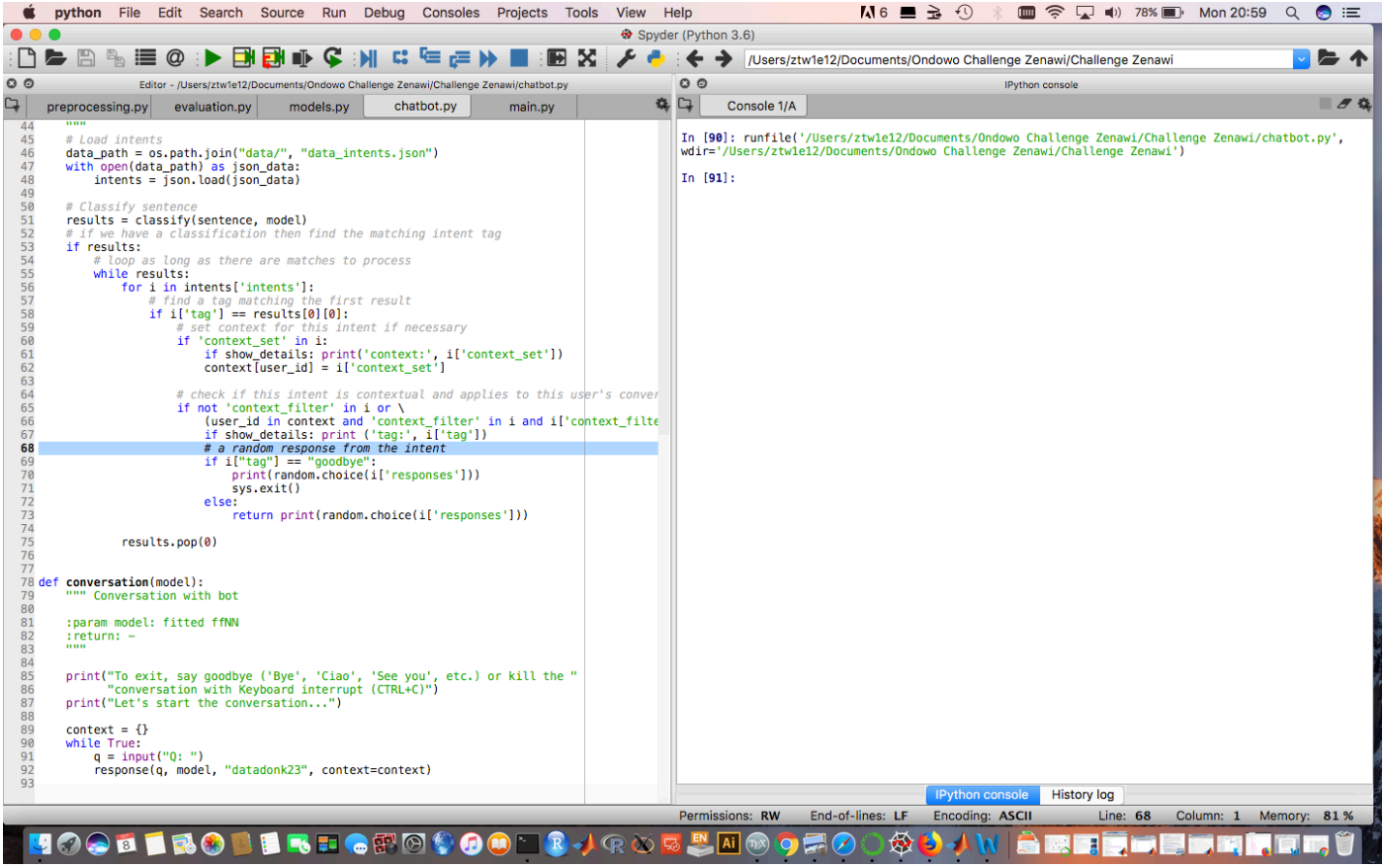

python File Edit Search Source Run Debug Consoles Projects Tools View Help
Mon 20:55
Spyder (Python 3.6)
/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi
Editor - /Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi/preprocessing.py
preprocessin...
evaluation.py
models.py
chatbot.py
main.py
data.
"""
preprocessing
NLP tasks
"""
import json, nltk, random, pickle, string
random.seed(23)
nltk.data.path.append("/home/<user>/nltk_data")
from nltk import punkt
from nltk.stem.lancaster import LancasterStemmer
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
def clean_sequence(seq):
""" Stemming, remove stopwords and punctuation from input sequence
:param seq: [word]
:return: [word] - cleaned seq of words
" seq is missing and I wonder how sould that be writen"
"""
stemmer = LancasterStemmer()
with open("data/custom_stopwords", "rb") as f:
stop_words = pickle.load(f)
cleaned_seq = [stemmer.stem(w.lower()) for w in seq if w not in
stop_words and w not in string.punctuation]
return cleaned_seq
def parse_training_data(path):
""" Parse training data and create dictonary for app
:param path: path to training data
:return: [word], [class], [document]
"""
words = []
classes = []
documents = []
with open("data/data_intents.json") as json_data:
intents = json.load(json_data)
IPython console
Console 1/A
In [85]:
In [85]: runfile('/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi/
preprocessing.py', wdir='/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi')
Reloaded modules: preprocessing, models, evaluation, utils, utils.tensorboard_utils,
utils.evaluation_utils, training, chatbot
In [86]:
IPython console
History log
Permissions: RW
End-of-lines: LF
Encoding: ASCII
Line: 59
Column: 1
Memory: 82 %

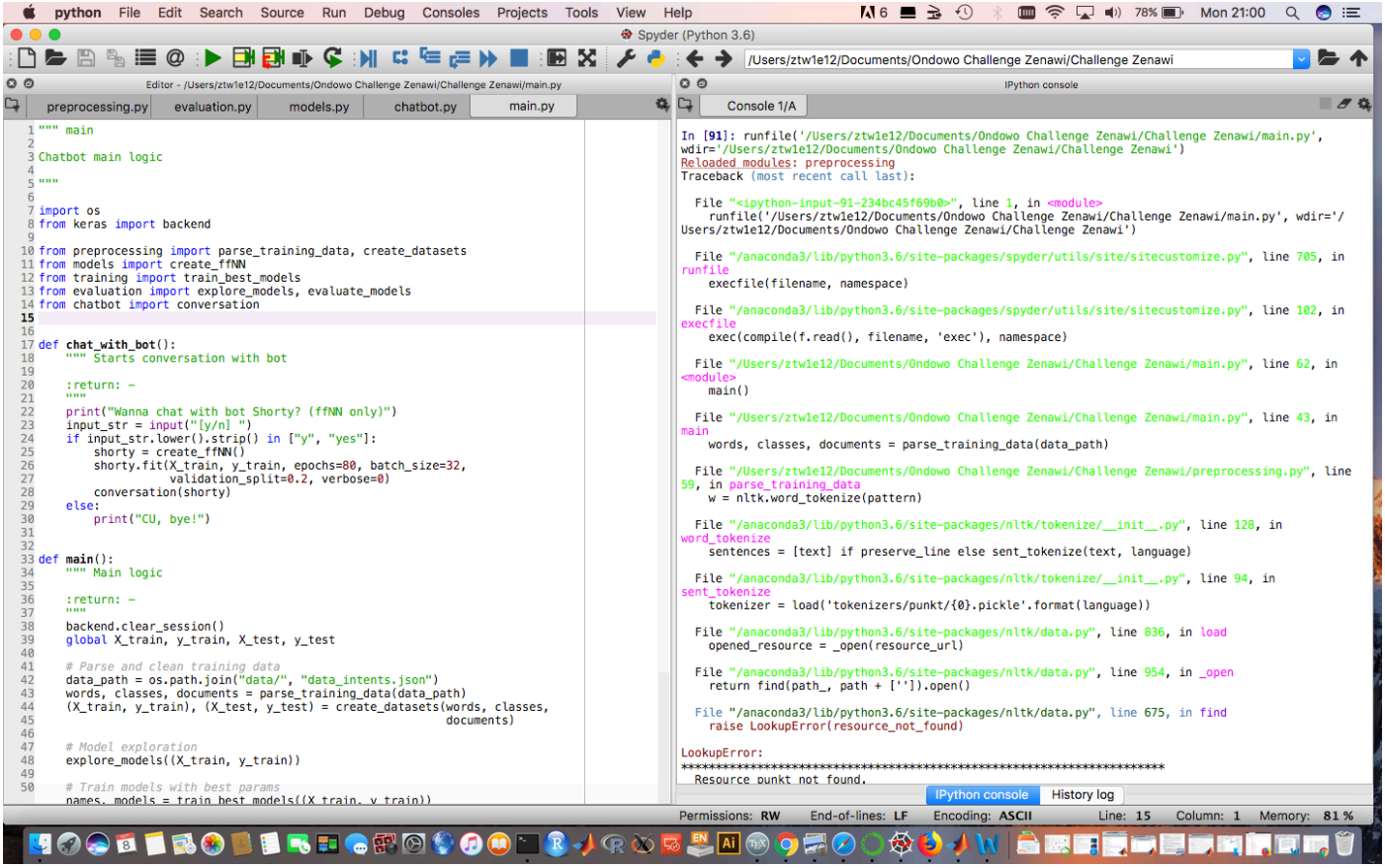

python File Edit Search Source Run Debug Consoles Projects Tools View Help
Mon 20:55
Spyder (Python 3.6)
/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi
Editor - /Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi/evaluation.py
preprocessin...
evaluation.py
models.py
chatbot.py
main.py
data.
:param model_names: [model_name]
:param trained_models: [trained_model]
:return: –
"""
(X_train, y_train) = trainset
(X_test, y_test) = testset
eval_results = {}
# Input reshaping
X_train_rec = X_train.reshape(114, 1, 105)
X_test_rec = X_test.reshape(13, 1, 105)
y_train_ensemble = np.argmax(y_train, axis=1)
y_test = np.argmax(y_test, axis=1)
for name, model in zip(model_names, trained_models):
if name in ["LSTM", "BiLSTM"]:
train_acc = model.score(X_train_rec, y_train)
y_pred = model.predict(X_test_rec)
elif (name == "ffNN"):
train_acc = model.score(X_train, y_train)
y_pred = model.predict(X_test)
else:
train_acc = model.score(X_train, y_train_ensemble)
y_pred = model.predict(X_test)
eval_results[name] = eval_clf(y_test, y_pred, train_acc)
eval = pd.DataFrame.from_dict(eval_results, orient="index",
columns=["Test_acc", "Precision",
"Recall", "FBeta",
"Train_acc"]).sort_values(
["Test_acc", "Train_acc"], ascending=False)
eval.to_csv("model/test_summary.csv")
print(eval)
class ModelExplorer:
""" Helper class for model selection
Template by David Batista, Panagiotis Katsaroumpas
(http://www.davidsbatista.net/blog/2018/02/23/model_optimization/)
"""
def __init__(self, models, params):
if not set(models.keys()).issubset(set(params.keys())):
missing_params = list(set(models.keys()) - set(params.keys()))
raise ValueError(
"Some estimators are missing parameters: %s" % missing_params)
self.models = models
self.params = params
self.keys = models.keys()
self.grid_searches = {}
IPython console
Console 1/A
In [86]: runfile('/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi/evaluation.py',
wdir='/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi')
In [87]:
IPython console
History log
Permissions: RW
End-of-lines: LF
Encoding: ASCII
Line: 23
Column: 8
Memory: 81 %

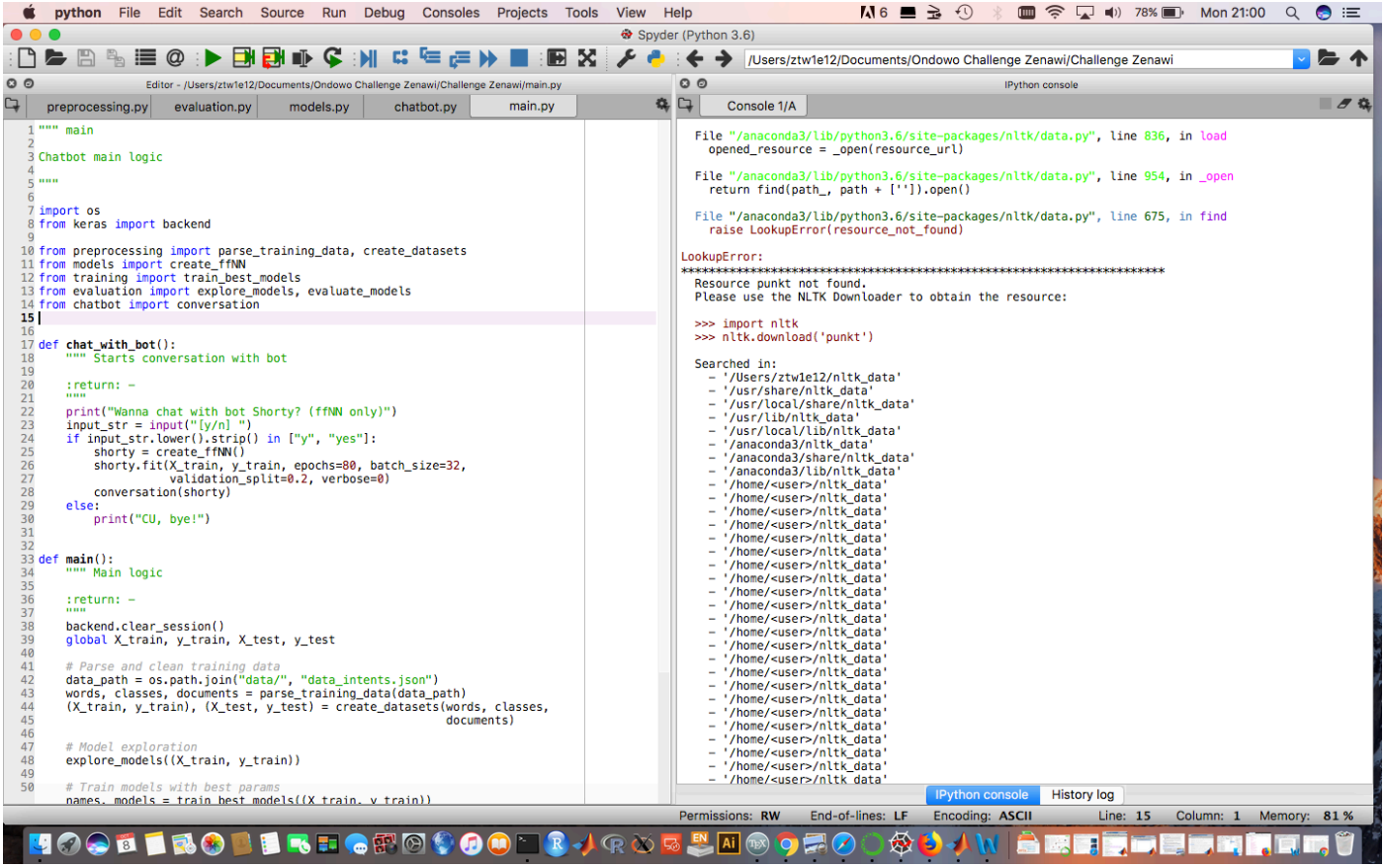


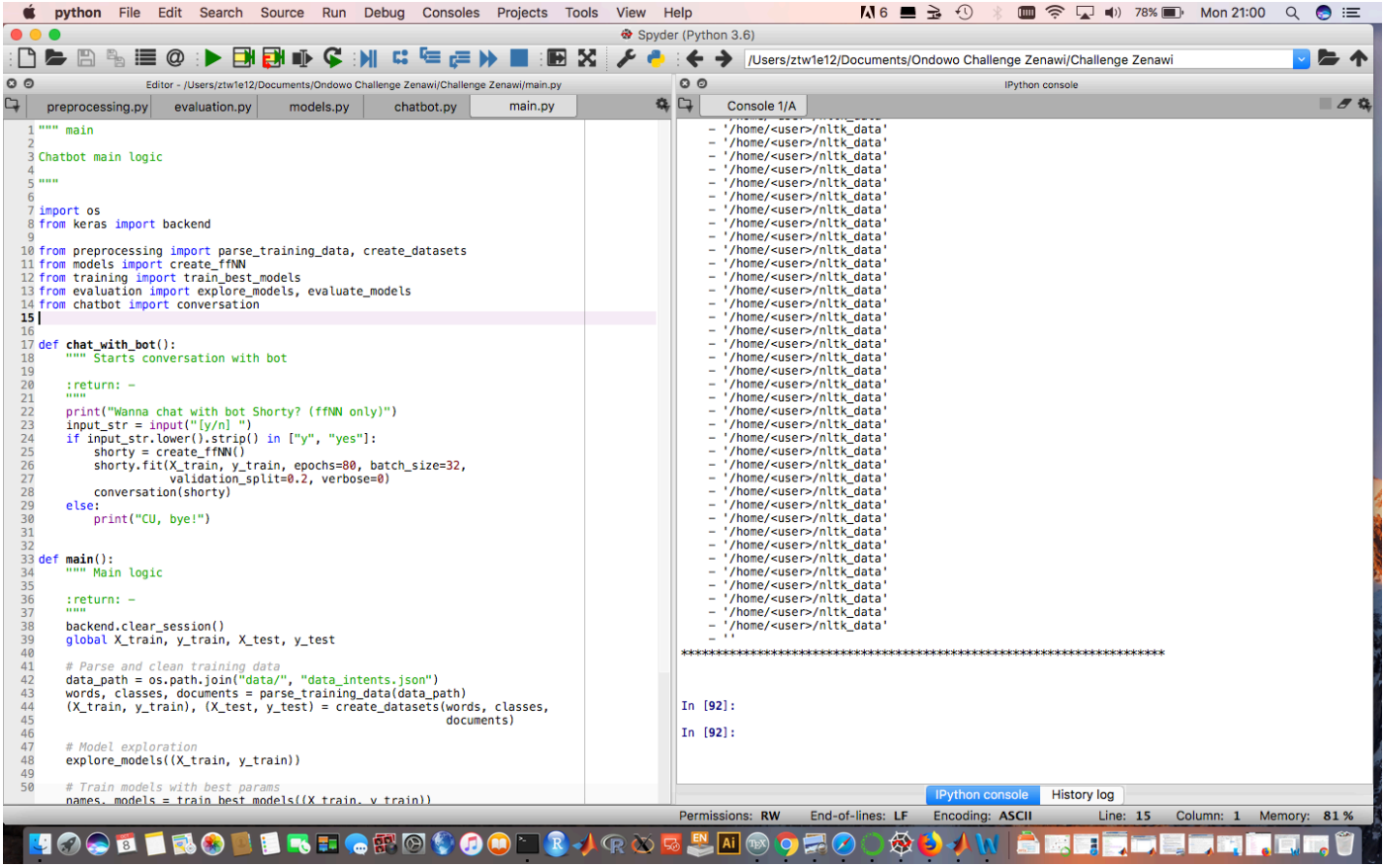

```
133     params = {
134         "Bagging": {"n_estimators": [15, 30], "max_features": [
135             1., 0.9, 0.8], "n_jobs": [1]},
136         "RandomForest": {"n_estimators": [10, 20],
137                          "criterion": ["gini", "entropy"],
138                          "max_features": ["sqrt", "log2"],
139                          "n_jobs": [-1]}
140     }
141
142     return (models, params)
143
144
145 def best_ensemble():
146     """ Ensemble models with best params taken from
147     model/ensembleCLFs_summary.csv
148
149     :return: zip("name", model)
150     """
151     names = ["Bagging", "RandomForest", "ExtraTrees", "AdaBoost",
152              "GradBoost", "Voting"]
153
154     # Params taken from model/ensembleCLF.csv
155     bagging_clf = BaggingClassifier(n_estimators=30, max_features=0.8,
156                                     random_state=23)
157     rf_clf = RandomForestClassifier(n_estimators=20, max_features="log2",
158                                     criterion="entropy", random_state=23)
159
160     return zip(names, [bagging_clf, rf_clf])
161
162
163 def stacking():
164     """ Stack best models into one CLF
165
166     :return: stacked_model
167     """
168     classifiers = []
169
170     # List of CLFs
171     for (_, model) in best_ensemble():
172         classifiers.append(model)
173     # Meta-learner
174     superlearner = RandomForestClassifier(random_state=23)
175
176     # Stack models
177     np.random.seed(23)
178     stacking_clf = StackingCVClassifier(classifiers,
179                                         meta_classifier=superlearner, cv=5)
180
181     return stacking_clf
182
```


python File Edit Search Source Run Debug Consoles Projects Tools View Help
Mon 20:56
Spyder (Python 3.6)
/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi
Editor - /Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi/chatbot.py
preprocessin...
evaluation.py
models.py
chatbot.py
main.py
data.
"""
# Load intents
data_path = os.path.join("data/", "data_intents.json")
with open(data_path) as json_data:
intents = json.load(json_data)
# Classify sentence
results = classify(sentence, model)
# if we have a classification then find the matching intent tag
if results:
# loop as long as there are matches to process
while results:
for i in intents['intents']:
# find a tag matching the first result
if i['tag'] == results[0][0]:
# set context for this intent if necessary
if 'context_set' in i:
if show_details: print('context:', i['context_set'])
context[user_id] = i['context_set']
# check if this intent is contextual and applies to this user's conver
if not 'context_filter' in i or \
(user_id in context and 'context_filter' in i and i['context_filte
if show_details: print ('tag:', i['tag'])
# a random response from the intent
if i["tag"] == "goodbye":
print(random.choice(i['responses']))
sys.exit()
else:
return print(random.choice(i['responses']))
results.pop(0)
def conversation(model):
""" Conversation with bot
:param model: fitted ffNN
:return: -
"""
print("To exit, say goodbye ('Bye', 'Ciao', 'See you', etc.) or kill the "
"conversation with Keyboard interrupt (CTRL+C)")
print("Let's start the conversation...")
context = {}
while True:
q = input("Q: ")
response(q, model, "datadonk23", context=context)
IPython console
Console 1/A
In [88]: runfile('/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi/chatbot.py',
wdir='/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi')
Reloaded modules: evaluation, utils, utils.tensorboard_utils, utils.evaluation_utils
In [89]:
IPython console
History log
Permissions: RW
End-of-lines: LF
Encoding: ASCII
Line: 68
Column: 1
Memory: 81 %


python File Edit Search Source Run Debug Consoles Projects Tools View Help
Mon 20:59
Spyder (Python 3.6)
/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi
Editor - /Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi/chatbot.py
preprocessing.py
evaluation.py
models.py
chatbot.py
main.py
IPython console
Console 1/A
In [90]: runfile('/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi/chatbot.py', wdir='/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi')
In [91]:
IPython console
History log
Permissions: RW
End-of-lines: LF
Encoding: ASCII
Line: 68
Column: 1
Memory: 81 %


python File Edit Search Source Run Debug Consoles Projects Tools View Help
Mon 21:00
Spyder (Python 3.6)
/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi
Editor - /Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi/main.py
preprocessing.py
evaluation.py
models.py
chatbot.py
main.py
"""
main
Chatbot main logic
"""
import os
from keras import backend
from preprocessing import parse_training_data, create_datasets
from models import create_ffNN
from training import train_best_models
from evaluation import explore_models, evaluate_models
from chatbot import conversation
def chat_with_bot():
""" Starts conversation with bot
:return: -
"""
print("Wanna chat with bot Shorty? (ffNN only)")
input_str = input("[y/n] ")
if input_str.lower().strip() in ["y", "yes"]:
shorty = create_ffNN()
shorty.fit(X_train, y_train, epochs=80, batch_size=32,
validation_split=0.2, verbose=0)
conversation(shorty)
else:
print("CU, bye!")
def main():
""" Main logic
:return: -
"""
backend.clear_session()
global X_train, y_train, X_test, y_test
# Parse and clean training data
data_path = os.path.join("data/", "data_intents.json")
words, classes, documents = parse_training_data(data_path)
(X_train, y_train), (X_test, y_test) = create_datasets(words, classes,
documents)
# Model exploration
explore_models((X_train, y_train))
# Train models with best params
IPython console
Console 1/A
In [91]: runfile('/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi/main.py',
wdir='/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi')
Reloaded modules: preprocessing
Traceback (most recent call last):
File "<ipython-input-91-234bc45f69b0>", line 1, in <module>
runfile('/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi/main.py', wdir='/
Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi')
File "/anaconda3/lib/python3.6/site-packages/spyder/utils/site/sitecustomize.py", line 705, in
runfile
execfile(filename, namespace)
File "/anaconda3/lib/python3.6/site-packages/spyder/utils/site/sitecustomize.py", line 102, in
execfile
exec(compile(f.read(), filename, 'exec'), namespace)
File "/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi/main.py", line 62, in
<module>
main()
File "/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi/main.py", line 43, in
main
words, classes, documents = parse_training_data(data_path)
File "/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi/preprocessing.py", line
59, in parse_training_data
w = nltk.word_tokenize(pattern)
File "/anaconda3/lib/python3.6/site-packages/nltk/tokenize/__init__.py", line 128, in
word_tokenize
sentences = [text] if preserve_line else sent_tokenize(text, language)
File "/anaconda3/lib/python3.6/site-packages/nltk/tokenize/__init__.py", line 94, in
sent_tokenize
tokenizer = load('tokenizers/punkt/{0}.pickle'.format(language))
File "/anaconda3/lib/python3.6/site-packages/nltk/data.py", line 836, in load
opened_resource = _open(resource_url)
File "/anaconda3/lib/python3.6/site-packages/nltk/data.py", line 954, in _open
return find(path_, path + ['']).open()
File "/anaconda3/lib/python3.6/site-packages/nltk/data.py", line 675, in find
raise LookupError(resource_not_found)
LookupError:
****
Resource punkt not found.
IPython console
History log
Permissions: RW
End-of-lines: LF
Encoding: ASCII
Line: 15
Column: 1
Memory: 81 %


python File Edit Search Source Run Debug Consoles Projects Tools View Help
Mon 21:00
Spyder (Python 3.6)
/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi
Editor - /Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi/main.py
preprocessing.py
evaluation.py
models.py
chatbot.py
main.py
"""
main
Chatbot main logic
"""
import os
from keras import backend
from preprocessing import parse_training_data, create_datasets
from models import create_ffNN
from training import train_best_models
from evaluation import explore_models, evaluate_models
from chatbot import conversation
def chat_with_bot():
""" Starts conversation with bot
:return: -
"""
print("Wanna chat with bot Shorty? (ffNN only)")
input_str = input("[y/n] ")
if input_str.lower().strip() in ["y", "yes"]:
shorty = create_ffNN()
shorty.fit(X_train, y_train, epochs=80, batch_size=32,
validation_split=0.2, verbose=0)
conversation(shorty)
else:
print("CU, bye!")
def main():
""" Main logic
:return: -
"""
backend.clear_session()
global X_train, y_train, X_test, y_test
# Parse and clean training data
data_path = os.path.join("data/", "data_intents.json")
words, classes, documents = parse_training_data(data_path)
(X_train, y_train), (X_test, y_test) = create_datasets(words, classes,
documents)
# Model exploration
explore_models((X_train, y_train))
# Train models with best params
IPython console
Console 1/A
File "/anaconda3/lib/python3.6/site-packages/nltk/data.py", line 836, in load
opened_resource = _open(resource_url)
File "/anaconda3/lib/python3.6/site-packages/nltk/data.py", line 954, in _open
return find(path_, path + ['']).open()
File "/anaconda3/lib/python3.6/site-packages/nltk/data.py", line 675, in find
raise LookupError(resource_not_found)
LookupError:
Resource punkt not found.
Please use the NLTK Downloader to obtain the resource:
>>> import nltk
>>> nltk.download('punkt')
Searched in:
- '/Users/ztw1e12/nltk_data'
- '/usr/share/nltk_data'
- '/usr/local/share/nltk_data'
- '/usr/lib/nltk_data'
- '/usr/local/lib/nltk_data'
- '/anaconda3/nltk_data'
- '/anaconda3/share/nltk_data'
- '/anaconda3/lib/nltk_data'
- '/home/<user>/nltk_data'
IPython console
History log
Permissions: RW
End-of-lines: LF
Encoding: ASCII
Line: 15
Column: 1
Memory: 81 %


python File Edit Search Source Run Debug Consoles Projects Tools View Help
Mon 21:00
Spyder (Python 3.6)
/Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi
Editor - /Users/ztw1e12/Documents/Ondowo Challenge Zenawi/Challenge Zenawi/main.py
preprocessing.py
evaluation.py
models.py
chatbot.py
main.py
"""" main
Chatbot main logic
"""
import os
from keras import backend
from preprocessing import parse_training_data, create_datasets
from models import create_ffNN
from training import train_best_models
from evaluation import explore_models, evaluate_models
from chatbot import conversation
def chat_with_bot():
""" Starts conversation with bot
:return: -
"""
print("Wanna chat with bot Shorty? (ffNN only)")
input_str = input("[y/n] ")
if input_str.lower().strip() in ["y", "yes"]:
shorty = create_ffNN()
shorty.fit(X_train, y_train, epochs=80, batch_size=32,
validation_split=0.2, verbose=0)
conversation(shorty)
else:
print("CU, bye!")
def main():
""" Main logic
:return: -
"""
backend.clear_session()
global X_train, y_train, X_test, y_test
# Parse and clean training data
data_path = os.path.join("data/", "data_intents.json")
words, classes, documents = parse_training_data(data_path)
(X_train, y_train), (X_test, y_test) = create_datasets(words, classes,
documents)
# Model exploration
explore_models((X_train, y_train))
# Train models with best params
IPython console
Console 1/A
- '/home/<user>/nltk_data'
In [92]:
In [92]:
IPython console
History log
Permissions: RW
End-of-lines: LF
Encoding: ASCII
Line: 15
Column: 1
Memory: 81 %