Python Notebook

{

"cells": [

{

"cell\_type": "markdown",

"metadata": {

"id": "f2S8I2ny-ovS"

},

"source": [

"# NLE Assignment: Sentiment Classification\n",

"\n",

"In this assignment, you will be investigating NLP methods for distinguishing positive and negative reviews written about movies.\n",

"\n",

"For assessment, you are expected to complete and submit this notebook file. When answers require code, you may import and use library functions (unless explicitly told otherwise). All of your own code should be included in the notebook rather than imported from elsewhere. Written answers should also be included in the notebook. You should insert as many extra cells as you want and change the type between code and markdown as appropriate.\n",

"\n",

"Marking guidelines are provided as a separate document.\n",

"\n",

"The first few cells contain code to set-up the assignment and bring in some data. In order to provide unique datasets for analysis by different students, you must enter your candidate number in the following cell. Otherwise do not change the code in these cells."

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "1gXQAZas-l9c"

},

"outputs": [],

"source": [

"candidateno=11111119 #this MUST be updated to your candidate number so that you get a unique data sample\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "nk8JTP88A8vs"

},

"outputs": [],

"source": [

"#do not change the code in this cell\n",

"#preliminary imports\n",

"\n",

"#set up nltk\n",

"import nltk\n",

"nltk.download('punkt')\n",

"nltk.download('stopwords')\n",

"nltk.download('movie\_reviews')\n",

"from nltk.corpus import stopwords\n",

"from nltk.tokenize import word\_tokenize\n",

"from nltk.corpus import movie\_reviews\n",

"\n",

"#for setting up training and testing data\n",

"import random\n",

"\n",

"#useful other tools\n",

"import re\n",

"import pandas as pd\n",

"import matplotlib.pyplot as plt\n",

"%matplotlib inline\n",

"from itertools import zip\_longest\n",

"from nltk.probability import FreqDist\n",

"from nltk.classify.api import ClassifierI\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "BHBkzAccCVaZ"

},

"outputs": [],

"source": [

"#do not change the code in this cell\n",

"def split\_data(data, ratio=0.7): # when the second argument is not given, it defaults to 0.7\n",

" \"\"\"\n",

" Given corpus generator and ratio:\n",

" - partitions the corpus into training data and test data, where the proportion in train is ratio,\n",

"\n",

" :param data: A corpus generator.\n",

" :param ratio: The proportion of training documents (default 0.7)\n",

" :return: a pair (tuple) of lists where the first element of the \n",

" pair is a list of the training data and the second is a list of the test data.\n",

" \"\"\"\n",

" \n",

" data = list(data) \n",

" n = len(data) \n",

" train\_indices = random.sample(range(n), int(n \* ratio)) \n",

" test\_indices = list(set(range(n)) - set(train\_indices)) \n",

" train = [data[i] for i in train\_indices] \n",

" test = [data[i] for i in test\_indices] \n",

" return (train, test) \n",

" \n",

"\n",

"def get\_train\_test\_data():\n",

" \n",

" #get ids of positive and negative movie reviews\n",

" pos\_review\_ids=movie\_reviews.fileids('pos')\n",

" neg\_review\_ids=movie\_reviews.fileids('neg')\n",

" \n",

" #split positive and negative data into training and testing sets\n",

" pos\_train\_ids, pos\_test\_ids = split\_data(pos\_review\_ids)\n",

" neg\_train\_ids, neg\_test\_ids = split\_data(neg\_review\_ids)\n",

" #add labels to the data and concatenate\n",

" training = [(movie\_reviews.words(f),'pos') for f in pos\_train\_ids]+[(movie\_reviews.words(f),'neg') for f in neg\_train\_ids]\n",

" testing = [(movie\_reviews.words(f),'pos') for f in pos\_test\_ids]+[(movie\_reviews.words(f),'neg') for f in neg\_test\_ids]\n",

" \n",

" return training, testing"

]

},

{

"cell\_type": "markdown",

"metadata": {

"id": "1N3LWwBYICPP"

},

"source": [

"When you have run the cell below, your unique training and testing samples will be stored in `training\_data` and `testing\_data`"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "HJLegkdPFUJA"

},

"outputs": [],

"source": [

"#do not change the code in this cell\n",

"random.seed(candidateno)\n",

"training\_data,testing\_data=get\_train\_test\_data()\n",

"print(\"The amount of training data is {}\".format(len(training\_data)))\n",

"print(\"The amount of testing data is {}\".format(len(testing\_data)))\n",

"print(\"The representation of a single data item is below\")\n",

"print(training\_data[0])"

]

},

{

"cell\_type": "markdown",

"metadata": {

"id": "VE3bKQbB50Rq"

},

"source": [

"1) \n",

"a) \*\*Generate\*\* a list of 10 content words which are representative of the positive reviews in your training data.\n",

"\n",

"b) \*\*Generate\*\* a list of 10 content words which are representative of the negative reviews in your training data.\n",

"\n",

"c) \*\*Explain\*\* what you have done and why\n",

"\n",

"[20\\%]"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "QkT9CWv250Rq"

},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "GQBi7JVR50Rr"

},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "OIqSAumd50Rs"

},

"outputs": [],

"source": []

},

{

"cell\_type": "markdown",

"metadata": {

"id": "Zljze\_ZJ50Rt"

},

"source": []

},

{

"cell\_type": "markdown",

"metadata": {

"id": "TApOQE6vND20"

},

"source": [

"2) \n",

"a) \*\*Use\*\* the lists generated in Q1 to build a \*\*word list classifier\*\* which will classify reviews as being positive or negative.\n",

"\n",

"b) \*\*Explain\*\* what you have done.\n",

"\n",

"[12.5\\%]\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "BThDMrcmODJy"

},

"outputs": [],

"source": []

},

{

"cell\_type": "markdown",

"metadata": {

"id": "B6xxvcyr50Rv"

},

"source": []

},

{

"cell\_type": "markdown",

"metadata": {

"id": "dL1iL9jg50Rv"

},

"source": [

"3)\n",

"a) \*\*Calculate\*\* the accuracy, precision, recall and F1 score of your classifier.\n",

"\n",

"b) Is it reasonable to evaluate the classifier in terms of its accuracy? \*\*Explain\*\* your answer and give a counter-example (a scenario where it would / would not be reasonable to evaluate the classifier in terms of its accuracy).\n",

"\n",

"[20\\%]"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "r1PrZnTe50Rw"

},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "Uq4Fmr8S50Rx"

},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "NJhRGovu50Ry"

},

"outputs": [],

"source": []

},

{

"cell\_type": "markdown",

"metadata": {

"id": "0gHYwj0i50Ry"

},

"source": []

},

{

"cell\_type": "markdown",

"metadata": {

"id": "LIS9UpmJNEAp"

},

"source": [

"4) \n",

"a) \*\*Construct\*\* a Naive Bayes classifier (e.g., from NLTK).\n",

"\n",

"b) \*\*Compare\*\* the performance of your word list classifier with the Naive Bayes classifier. \*\*Discuss\*\* your results. \n",

"\n",

"[12.5\\%]"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "hG4DSeqD50Rz"

},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "56\_oD8ET50Rz"

},

"outputs": [],

"source": []

},

{

"cell\_type": "markdown",

"metadata": {

"id": "xfuYer9U50Rz"

},

"source": []

},

{

"cell\_type": "markdown",

"metadata": {

"id": "LGDXaVDqOSfY"

},

"source": [

"5) \n",

"a) Design and \*\*carry out an experiment\*\* into the impact of the \*\*length of the wordlists\*\* on the wordlist classifier. Make sure you \*\*describe\*\* design decisions in your experiment, include a \*\*graph\*\* of your results and \*\*discuss\*\* your conclusions. \n",

"\n",

"b) Would you \*\*recommend\*\* a wordlist classifier or a Naive Bayes classifier for future work in this area? \*\*Justify\*\* your answer.\n",

"\n",

"[25\\%]\n"

]

},

{

"cell\_type": "markdown",

"metadata": {

"id": "5PVBBPhh50R0"

},

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "9tBhqZtw50R1"

},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "J1ozKYMa50R1"

},

"outputs": [],

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "vzN\_HIuC50R1"

},

"outputs": [],

"source": []

},

{

"cell\_type": "markdown",

"metadata": {

"id": "nYZVd15L50R2"

},

"source": []

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 34

},

"id": "34rdlS\_iPov6",

"outputId": "06237f84-098e-4c36-efc5-52ab25d68da5"

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Submission length is 0\n"

]

}

],

"source": [

"##This code will word count all of the markdown cells in the notebook saved at filepath\n",

"##Running it before providing any answers shows that the questions have a word count of 437\n",

"\n",

"import io\n",

"from nbformat import current\n",

"\n",

"#filepath=\"/content/drive/My Drive/NLE Notebooks/assessment/assignment1.ipynb\"\n",

"filepath=\"NLassignment2021.ipynb\"\n",

"question\_count=437\n",

"\n",

"with io.open(filepath, 'r', encoding='utf-8') as f:\n",

" nb = current.read(f, 'json')\n",

"\n",

"word\_count = 0\n",

"for cell in nb.worksheets[0].cells:\n",

" if cell.cell\_type == \"markdown\":\n",

" word\_count += len(cell['source'].replace('#', '').lstrip().split(' '))\n",

"print(\"Submission length is {}\".format(word\_count-question\_count))"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "PtqCcG6wPsmf"

},

"outputs": [],

"source": []

}

],

"metadata": {

"colab": {

"provenance": []

},

"kernelspec": {

"display\_name": "Python 3",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.7.4"

}

},

"nbformat": 4,

"nbformat\_minor": 0

}