{

"cells": [

{

"cell\_type": "code",

"execution\_count": 1,

"metadata": {},

"outputs": [],

"source": [

"import numpy as np\n",

"import pandas as pd\n",

"import os\n",

"import re\n",

"import string\n",

"from textblob import TextBlob\n",

"from textblob.sentiments import NaiveBayesAnalyzer\n",

"from sklearn.feature\_extraction.text import CountVectorizer\n",

"from sklearn.linear\_model import LogisticRegression\n",

"from sklearn.metrics import accuracy\_score\n",

"from sklearn.model\_selection import train\_test\_split"

]

},

{

"cell\_type": "code",

"execution\_count": 2,

"metadata": {},

"outputs": [],

"source": [

"#input text-tweets\n",

"train = pd.read\_csv('Dataset\_sarcasm.csv', encoding='Latin-1')"

]

},

{

"cell\_type": "code",

"execution\_count": 3,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"'neg'"

]

},

"execution\_count": 3,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"def remove\_pattern(input\_txt, pattern):\n",

" r = re.findall(pattern, input\_txt)\n",

" for i in r:\n",

" input\_txt = re.sub(i, '', input\_txt)\n",

" \n",

" return input\_txt \n",

"\n",

"#remove usernames\n",

"train['tweet'] = np.vectorize(remove\_pattern)(train['tweet'], \"@[\\w]\*\") \n",

"#remove Special characters\n",

"train['tweet'] = train['tweet'].str.replace(\"[^a-zA-Z#]\", \" \")\n",

"#remove short words\n",

"train['tweet'] = train['tweet'].apply(lambda x: ' '.join([w for w in x.split() if len(w)>3]))\n",

"\n",

"text = ['saying a tweet is sarcastic is sarcasm itself']\n",

"\n",

"blob = TextBlob(text[0], analyzer=NaiveBayesAnalyzer())\n",

"blob.sentiment.classification"

]

},

{

"cell\_type": "code",

"execution\_count": 4,

"metadata": {},

"outputs": [],

"source": [

"from nltk.corpus import stopwords\n",

"\n",

"english\_stop\_words = stopwords.words('english')\n",

"def remove\_stop\_words(corpus):\n",

" removed\_stop\_words = []\n",

" for review in corpus:\n",

" removed\_stop\_words.append(\n",

" ' '.join([word for word in review.split() \n",

" if word not in english\_stop\_words])\n",

" )\n",

" return removed\_stop\_words\n",

"\n",

"no\_stop\_words\_train = pd.DataFrame(columns=['tweet'])\n",

"no\_stop\_words\_train['tweet']= remove\_stop\_words(train['tweet'])\n",

"#print(no\_stop\_words\_train['tweet'])"

]

},

{

"cell\_type": "code",

"execution\_count": 5,

"metadata": {},

"outputs": [],

"source": [

"def get\_lemmatized\_text(corpus):\n",

" \n",

" from nltk.stem import WordNetLemmatizer\n",

" lemmatizer = WordNetLemmatizer()\n",

" return [' '.join([lemmatizer.lemmatize(word) for word in review.split()]) for review in corpus]\n",

"\n",

"lemmatized\_train = pd.DataFrame(columns=['tweet'])\n",

"lemmatized\_train['tweet'] = get\_lemmatized\_text(no\_stop\_words\_train['tweet'])"

]

},

{

"cell\_type": "code",

"execution\_count": 6,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Accuracy for C=0.001: 0.9976872800402212\n",

"Accuracy for C=0.005: 0.9976872800402212\n",

"Accuracy for C=0.01: 0.9976872800402212\n",

"Accuracy for C=0.05: 0.9976872800402212\n",

"Accuracy for C=0.1: 0.9976872800402212\n"

]

},

{

"data": {

"text/plain": [

"LinearSVC(C=0.01, class\_weight=None, dual=True, fit\_intercept=True,\n",

" intercept\_scaling=1, loss='squared\_hinge', max\_iter=1000,\n",

" multi\_class='ovr', penalty='l2', random\_state=None, tol=0.0001,\n",

" verbose=0)"

]

},

"execution\_count": 6,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"from sklearn.feature\_extraction.text import CountVectorizer\n",

"from sklearn.model\_selection import train\_test\_split\n",

"from sklearn.metrics import accuracy\_score\n",

"from sklearn.svm import LinearSVC\n",

"\n",

"\n",

"stop\_words = ['in', 'of', 'at', 'a', 'the']\n",

"ngram\_vectorizer = CountVectorizer(binary=True, ngram\_range=(1, 3), stop\_words=stop\_words)\n",

"ngram\_vectorizer.fit(lemmatized\_train['tweet'])\n",

"X = ngram\_vectorizer.transform(lemmatized\_train['tweet'])\n",

"\n",

"X\_train, X\_val, y\_train, y\_val = train\_test\_split(\n",

" X, train['label'], train\_size = 0.75\n",

")\n",

"\n",

"for c in [0.001, 0.005, 0.01, 0.05, 0.1]:\n",

" \n",

" svm = LinearSVC(C=c)\n",

" svm.fit(X\_train, y\_train)\n",

" print (\"Accuracy for C=%s: %s\" \n",

" % (c, accuracy\_score(y\_val, svm.predict(X\_val))))\n",

" \n",

"# Accuracy for C=0.001: 0.88784\n",

"# Accuracy for C=0.005: 0.89456\n",

"# Accuracy for C=0.01: 0.89376\n",

"# Accuracy for C=0.05: 0.89264\n",

"# Accuracy for C=0.1: 0.8928\n",

" \n",

"final = LinearSVC(C=0.01)\n",

"final.fit(X, train['label'])"

]

},

{

"cell\_type": "code",

"execution\_count": 7,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"[1]\n"

]

}

],

"source": [

"# Final Accuracy: 0.90064\n",

"transform\_text = ngram\_vectorizer.transform(text)\n",

"print(final.predict(transform\_text))"

]

}

],

"metadata": {

"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": 2

}