

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Volume 4, Issue 6, April 2024

Text Classification in Natural Language Processing

Venkata Mahesh Babu Batta

https://orcid.org/0000-0002-1029-6402 M.Tech, Department of CSE University College of Engineering, Osmania University, Hyderabad, Telangana, India

Abstract: This paper presents an overview of text classification techniques, focusing on the pre-processing steps, feature extraction methods, and model selection strategies employed in the process. Algorithms such as Naive Bayes, Support Vector Machines (SVM), logistic regression, and neural networks are used. Furthermore, recent advancements in deep learning models for text classification, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are used. Comprehensive understanding of text classification methodologies in NLP and insights into current trends and challenges in the field are mentioned

Keywords: Natural Language Processing(NLP), Python

I. INTRODUCTION

Naive Bayes algorithm using Python: Use the popular scikit-learn library #python from sklearn.datasets import fetch_20newsgroups from sklearn.feature_extraction.text import TfidfVectorizer from sklearn.naive_bayes import MultinomialNB from sklearn.pipeline import make_pipeline from sklearn.metrics import classification_report, accuracy_score from sklearn.model_selection import train_test_split

Load the 20 newsgroups dataset (a collection of newsgroup documents)
data = fetch_20newsgroups(subset='all', shuffle=True, random_state=42)

Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.25, random_state=42)

Create a pipeline with TF-IDF vectorizer and Naive Bayes classifier model = make_pipeline(TfidfVectorizer(), MultinomialNB())

Train the model on the training data model.fit(X_train, y_train)

Predict the labels for the test set
y_pred = model.predict(X_test)

Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=data.target_names))

Copyright to IJARSCT www.ijarsct.co.in

DOI: 10.48175/IJARSCT-17645



282



International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Volume 4, Issue 6, April 2024

Explanation:

- start by importing necessary modules from scikit-learn.
- load the 20 newsgroups dataset using `fetch_20newsgroups()` function.
- split the dataset into training and testing sets using `train_test_split()` function.
- create a pipeline consisting of a TF-IDF vectorizer and a Multinomial Naive Bayes classifier using `make_pipeline()` function.
- train the model on the training data using `fit()` method.
- make predictions on the test data using `predict()` method.
- evaluate the model's performance by calculating accuracy and generating a classification report using `accuracy_score()` and `classification_report()` functions respectively.

Command Prompt-python Nam crosoft Windows [Versic) Microsoft Corporation	n 10.0.1904										- 0	>
) Microsoft Corporation												
(USELS (Maries / Cu C. (USEL	s (manes (bow	110803 (001	ĸ									
\Users\mahes\Downloads\ curacy: 0.8425297113752		NaiveBaye	s.py									
assification Report:												
	precision	recall	f1-score	support								
alt.atheism	0.88	0.72	0.79	198								
comp.graphics	0.86	0.79	0.82	245								
omp.os.ms-windows.misc	0.88	0.83	0.85	242								
np.sys.ibm.pc.hardware	0.66	0.86	0.75	238								
comp.sys.mac.hardware	0.95	0.84	0.89	250								
comp.windows.x	0.96	0.80	0.87	260								
misc.forsale	0.96	0.66	0.78	241								
rec.autos	0.89	0.93	0.91	244								
rec.motorcycles	0.91	0.95	0.93	219								
rec.sport.baseball	0.96	0.94	0.95	261								
rec.sport.hockey	0.90	0.98	0.94	245								
sci.crypt	0.78	0.98	0.87	251								
sci.electronics	0.92	0.80	0.85	249								
sci.med	0.97	0.88	0.92	249								
sci.space oc.religion.christian	0.88 0.49	0.98	0.93	240 245								
talk.politics.guns	0.49	0.99	0.00	245								
talk.politics.mideast	0.93	0.95	0.85	230								
talk.politics.misc	1.00	0.57	0.73	207								
talk.religion.misc	1.00	0.16	0.28	164								
cark.reiigion.misc	1.00	0.10	0.20	104								
accuracy			0.84	4712								
macro avg	0.88	0.83	0.83	4712								
weighted avg	0.88	0.84	0.84	4712								
										Activate Window		
										Go to Settings to activ	ate Windows	
	12	0.		-	-		-				8-71 PM	
P Type here to searchere	ch 🗧	🍋 🚊		a 💽	<u></u>	- 49	🧿 🔚	1 🧿		💙 36°C \land 🖻 🌾	24-Apr-24	. 5
		and the second se									24-Apt-24	

Support Vector Machines:

Text classification using Support Vector Machines (SVM) in Python with the scikit-learn library. use the 20 newsgroups dataset

#python

from sklearn.datasets import fetch_20newsgroups from sklearn.feature_extraction.text import TfidfVectorizer from sklearn.svm import SVC from sklearn.pipeline import make_pipeline from sklearn.metrics import classification_report, accuracy_score from sklearn.model_selection import train_test_split

Load the 20 newsgroups dataset
data = fetch_20newsgroups(subset='all', shuffle=True, random_state=42)

Copyright to IJARSCT www.ijarsct.co.in





International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Volume 4, Issue 6, April 2024

Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.25, random_state=42)

Create a pipeline with TF-IDF vectorizer and Support Vector Machines classifier model = make_pipeline(TfidfVectorizer(), SVC(kernel='linear'))

Train the model on the training data model.fit(X_train, y_train)

Predict the labels for the test set
y_pred = model.predict(X_test)

Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification report(y test, y pred, target names=data.target names))

Explanation:

- import necessary modules from scikit-learn.
- load the 20 newsgroups dataset using `fetch_20newsgroups()` function.
- split the dataset into training and testing sets using `train_test_split()` function.
- create a pipeline consisting of a TF-IDF vectorizer and a Support Vector Machines classifier using `make_pipeline()` function.
- train the model on the training data using `fit()` method.
- make predictions on the test data using `predict()` method.
- evaluate the model's performance by calculating accuracy and generating a classification report using `accuracy_score()` and `classification_report()` functions respectively.

Figure: Support Vector Machines

Logistic Regression:

Text classification using logistic regression in Python with scikit-learn, using the 20 newsgroups dataset.

#python

from sklearn.datasets import fetch_20newsgroups from sklearn.feature_extraction.text import TfidfVectorizer from sklearn.linear_model import LogisticRegression from sklearn.pipeline import make_pipeline from sklearn.metrics import classification_report, accuracy_score from sklearn.model selection import train test split

Load the 20 newsgroups dataset
data = fetch_20newsgroups(subset='all', shuffle=True, random_state=42)

Split the dataset into training and testing sets X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.25, random_state=42)

Copyright to IJARSCT www.ijarsct.co.in





International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Volume 4, Issue 6, April 2024

Create a pipeline with TF-IDF vectorizer and Logistic Regression classifier model = make_pipeline(TfidfVectorizer(), LogisticRegression(max_iter=1000))

Train the model on the training data model.fit(X_train, y_train)

Predict the labels for the test set
y_pred = model.predict(X_test)

Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=data.target_names))

Explanation:

- import necessary modules from scikit-learn.

- load the 20 newsgroups dataset using `fetch_20newsgroups()` function.

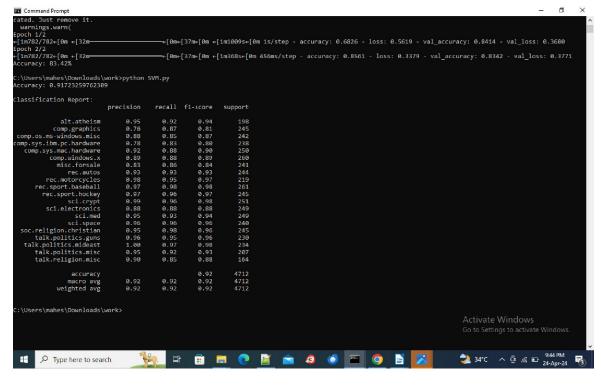
- split the dataset into training and testing sets using `train_test_split()` function.

-create a pipeline consisting of a TF-IDF vectorizer and a Logistic Regression classifier using `make_pipeline()` function.

- train the model on the training data using `fit()` method.

- make predictions on the test data using `predict()` method.

- evaluate the model's performance by calculating accuracy and generating a classification report using `accuracy_score()` and `classification_report()` functions respectively.



Copyright to IJARSCT www.ijarsct.co.in DOI: 10.48175/IJARSCT-17645



285



International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Volume 4, Issue 6, April 2024

Jsers\mahes>cd C:\User	s\mahes\Dowr	loads\wor	k							
Jsers\mahes\Downloads\		LogisticR	egression.	у						
iracy: 0.8921901528013	583									
sification Report:	precision	pocal1	f1-score	support						
	precision	recall	11-Score	Support						
alt.atheism	0,90	0.89	0.90	198						
comp.graphics	0.76	0.85	0.81	245						
p.os.ms-windows.misc	0.83	0.86	0.84	242						
.sys.ibm.pc.hardware	0.76	0.77	0.76	238						
omp.sys.mac.hardware	0.88	0.84	0.86	250						
comp.windows.x	0.90	0.86	0.88	260						
misc.forsale	0.76	0.83	0.80	241						
rec.autos	0.93	0.92	0.93	244						
rec.motorcycles	0.96	0.94	0.95	219						
rec.sport.baseball	0.96	0.97	0.97	261						
rec.sport.hockey sci.crypt	0.96	0.96	0.96	245 251						
sci.electronics	0.86	0.86	0.86	249						
sci.electronics sci.med	0.00	0.00	0.00	249						
sci.space	0.95	0.92	0.95	249						
c.religion.christian	0.87	0.96	0.94	246						
talk.politics.guns	0.92	0.90	0.92	230						
alk.politics.mideast	0.92	0.98	0.92	236						
talk.politics.misc	0.98	0.86	0.89	207						
talk.religion.misc	0.87	0.65	0.74	164						
accuracy			0.89	4712						
macro avg	0.89	0.89	0.89	4712						
weighted avg	0.89	0.89	0.89	4712						
Jsers\mahes\Downloads\	work>									
								Activate V	lindows	
								Go to Setting		
						_		💙 34°C 🗸		

Figure: Logistic Regression

Conventional Neural Networks:

Text classification using a conventional neural network (also known as a feedforward neural network or multilayer perceptron) in Python with Keras. use the 20 newsgroups dataset as before.

#python

import numpy as np
from sklearn.datasets import fetch_20newsgroups
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.utils import to_categorical

Load the 20 newsgroups dataset
data = fetch_20newsgroups(subset='all', shuffle=True, random_state=42)

Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.25, random_state=42)

Convert text data to TF-IDF vectors
vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

Copyright to IJARSCT www.ijarsct.co.in DOI: 10.48175/IJARSCT-17645



286



International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Volume 4, Issue 6, April 2024

Convert target labels to one-hot encoding num_classes = len(np.unique(y_train)) y_train_onehot = to_categorical(y_train, num_classes) y_test_onehot = to_categorical(y_test, num_classes)

Build a conventional neural network model model = Sequential() model.add(Dense(512, input_shape=(X_train_tfidf.shape[1],), activation='relu')) model.add(Dropout(0.5)) model.add(Dense(256, activation='relu')) model.add(Dropout(0.5)) model.add(Dense(num_classes, activation='softmax'))

Compile the model model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

Train the model model.fit(X_train_tfidf, y_train_onehot, epochs=5, batch_size=32, validation_split=0.1)

Evaluate the model
loss, accuracy = model.evaluate(X_test_tfidf, y_test_onehot)
print('Test Loss:', loss)
print('Test Accuracy:', accuracy)

Explanation:

- import necessary modules from scikit-learn and Keras.
- load the 20 newsgroups dataset using `fetch_20newsgroups()` function.
- split the dataset into training and testing sets using `train_test_split()` function.
- convert text data to TF-IDF vectors using `TfidfVectorizer()` from scikit-learn.
- convert target labels to one-hot encoding using `to_categorical()` from Keras.
- build a conventional neural network model using `Sequential()` from Keras and add dense layers with ReLU activation and dropout for regularization.
- compile the model with categorical crossentropy loss and Adam optimizer.
- train the model on the training data using `fit()` method.
- evaluate the model's performance on the test data using `evaluate()` method.





International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Volume 4, Issue 6, April 2024

Command Prompt - python CNN.py											
											٥
soc.religion.christian	0.87	0.96	0.91	245							
talk.politics.guns	0.92	0.93	0.92	230							
talk.politics.mideast	0.98	0.98	0.98	234							
talk.politics.misc	0.93	0.86	0.89	207							
talk.religion.misc	0.87	0.65	0.74	164							
accuracy			0.89	4712							
macro avg	0.89	0.89	0.89	4712							
weighted avg	0.89	0.89	0.89	4712							
Jsers\mahes\Downloads\work											
-04-24 20:43:33.077257: 1	tensorflo	w/core/uti	1/port.co	:113] oneDNM	I custom oper	ations are on.	You may see s	lightly di	fferent numerical	results du	e to fl
-point round-off errors f											
-04-24 20:43:57.192808: 1										results du	e to +1
-point round-off errors f sers\mahes\AppData\Local\	rom aitter	ent comput	ation ord	ers. To turn	them off, si	et the environ	ment variable	TF_ENABLE_	ONEDNN_OPTS=0		
argument to a layer. Whe										inpuc_snape	-/ Inpr
per(). init (activity r	equierizer	-activity	regularia	erer using a	a) Input(Sna	pey object as	the trist lay	er in the n	ioder instead.		
h 1/5	egoral izel.	-accivicy_	i eBorai 12	er, kwarga	<i>,</i>						
398/398⊷[0m ←[32m				m41956 FAm 90	Bms/sten = a	couracy: 0 409	+ - loss - 2 14	74 - val a	curacy: 0.9059 -	val loss.	8 3673
1 2/5											
398/398+[0m +[32m			'm⊷[0m ⊷[1	m284s←[0m 70	06ms/step - a	ccuracy: 0.963	1 - loss: 0.16	22 - val ad	curacy: 0.9208 -	val loss:	0.2770
h 3/5											
398/398+[0m +[32m		-+[0m+[37	m+[0m +[1	m312s+[0m 78	2ms/step - a	ccuracy: 0.994	6 - loss: 0.02	96 - val_ac	curacy: 0.9257 -	val_loss:	0.2729
h 4/5											
398/398+[0m +[32m			'm+[0m +[1	m546s←[0m 1s	/step - accu	racy: 0.9979 -	loss: 0.0152	 val_accur 	nacy: 0.9250 - va	1_loss: 0.2	
h 5/5						A Contractor of the				Maria and a second of a	
398/398←[0m +[32m 148/148←[0m +[32m						ccuracy: 0.998 racy: 0.9203 -		76 - val_ac	curacy: 0.9272 -	val_loss:	0.2881
Loss: 0.3083120584487915	-	-+[600+[33	melom el t	mozefom sous	/scep - accu	uarh: 018502 -	1055: 0.5424				
	2										
	3208										
	3208										
	3208										
	3208										
	3208										
	3208										
	3208										
	3208										
	3208										
	3208								Activate Mir		
	3208								Activate Wir		
	3208								Activate Wir Go to Settings ta		
	3208										
C 053: 0.30531203448/913 Accuracy: 0.928056001663	3208										
Accuracy: 0.92805600166	3208 100	1	a -							o activate Wi	20 P M
Accuracy: 0.92805600166	3208	L H	Ē .	1 🤁 🛓	Y 💼 4) (6) 🖻	9 2			o activate Wi	

Figure: Conventional Neural Networks

Recurrent Neural Networks(RNNs)

Text classification using Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) networks, in Python with Keras.

Use the IMDB movie review dataset for sentiment analysis.

#python

import numpy as np from keras.datasets import imdb from keras.models import Sequential from keras.layers import Embedding, LSTM, Dense from keras.preprocessing.sequence import pad_sequences

Set parameters max_features = 5000 # Number of words to consider as features maxlen = 400 # Cut texts after this number of words batch_size = 32 embedding_dims = 50 epochs = 2 # Increase this value for better accuracy

Load the data
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
Pad sequences to make them uniform length
x_train = pad_sequences(x_train, maxlen=maxlen)
x_test = pad_sequences(x_test, maxlen=maxlen)

Define the model
model = Sequential()

Copyright to IJARSCT www.ijarsct.co.in





International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Volume 4, Issue 6, April 2024

Embedding layer model.add(Embedding(max features, embedding dims, input length=maxlen))

LSTM layer
model.add(LSTM(100)) # You can adjust the number of LSTM units as per your requirement

Output layer
model.add(Dense(1, activation='sigmoid'))

Compile the model model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

Train the model model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validation data=(x test, y test))

Evaluate the model
scores = model.evaluate(x_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1] * 100))

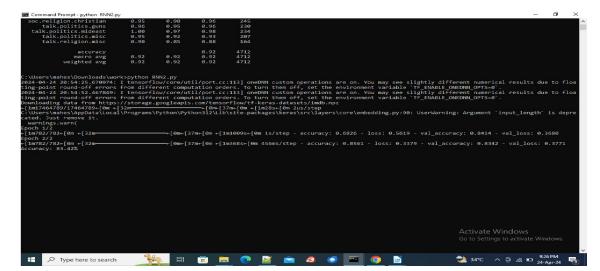


Figure: Recurrent Neural Networks

II. CONCLUSION

In conclusion, text classification stands as a cornerstone in Natural Language Processing (NLP), serving as a pivotal tool for organizing, categorizing, and understanding textual data at scale.

Traditional machine learning algorithms like Naive Bayes, Support Vector Machines (SVM), and Decision Trees have long been the cornerstone of text classification tasks, offering interpretable models and decent performance across various datasets. However, with the advent of deep learning, particularly Convolutional Networks (CNNs),

Copyright to IJARSCT www.ijarsct.co.in





International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Volume 4, Issue 6, April 2024

Recurrent Neural Networks (RNNs), and their variants like LSTM and GRU, text classification has seen remarkable advancements in accuracy and scalability.

Moreover, transfer learning techniques leveraging pre-trained language models, such as BERT, GPT, and their derivatives, have revolutionized text classification by providing models with rich contextual understanding and the ability to generalize across diverse domains with minimal task-specific training data.

REFERENCES:

- [1]. Jurafsky, D., & Martin, J. H. (2019). Speech and Language Processing (3rd ed.). Pearson.
- [2]. Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.
- [3]. Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media.
- [4]. Goldberg, Y. (2016). A Primer on Neural Network Models for Natural Language Processing. Journal of Artificial Intelligence Research, 57, 345-420.
- [5]. Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent Trends in Deep Learning Based Natural Language Processing. IEEE Computational Intelligence Magazine, 13(3), 55-75.
- [6]. Zhang, Y., & Wallace, B. (2017). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. arXiv preprint arXiv:1510.03820.
- [7]. Vaswani, A., et al. (2017). Attention Is All You Need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS).
- [8]. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805

