Fakulta elektrotechniky a informatiky

Katedra kybernetiky a umelej inteligencie Predmet :

SaSW

2022 / 2023

Zadanie:

Projekt Troll Detection

Kľúčove slová: Python, LSTM, GRU, BiLSTM, BiGRU

Spracovali:

David Lacko, Dmytro Lahunov, Dmytro Furman, Bohdan Tanasov

KKUI
Katedra kybernetiky
a umelej inteligencie

1 Úvod

Táto dokumentácia slúži ako pomôcka na pochopenie obsahu úlohy Troll detection a podrobnejšie popisuje vybrané postupy a fungovanie jednotlivých modelov úlohy.

1.1 Popis ulohy

Úlohou nášho tímu v tejto úlohe bolo vytvoriť modely, ktorý dokážu klasifikovať písaný text a presne určiť, čo je trolling a urážky.

1.2 Inšpirácia

Hlavným faktorom, prečo sme sa v tomto projekte vybrali je ze z veku informácií sme sa dostali do veku dezinformácií, urážlivých prejavov, trollingu, falošných správ alebo recenzií atď. Všetky tieto typy protispoločenského správania ovplyvňujú demokraciu v mnohých krajinách a prispievajú k polarizácii spoločnosti. Mnohokrát ide o trolling a šírenie toxických príspevkov, ktoré sa snažia manipulovať s názormi používateľov hľadajúcich odpovede v online priestore. Informačné technológie tak môžu spôsobiť masový chaos, hrubosť, nedostatok dôvery, osamelosť, polarizáciu spoločnosti, nabúranie volieb a iných demokratických postupov a väčší populizmus. Táto téma je v dnešnom svete veľmi dôležitá a my osobne máme záujem podieľať sa na možnom budúcom prelomovom vývoji v tejto oblasti.

2 Technické prostriedky

Na písanie kódu modelov sme použili programovací jazyk Python. Python je vysokoúrovňový univerzálny programovací jazyk s dynamickým striktným typovaním a automatickou správou pamäte zameraný na zvýšenie produktivity vývojárov, čitateľnosti a kvality kódu a zabezpečenie prenosnosti programov v ňom napísaných. Python je multiparadigmatický programovací jazyk, ktorý podporuje imperatívne, procedurálne, štrukturálne, objektovo orientované programovanie, metaprogramovanie a funkcionálne programovanie. Zovšeobecnené úlohy programovania sa riešia dynamickým typovaním. Aspektovo orientované programovanie čiastočne podporujú dekorátory, plnšiu podporu poskytujú ďalšie

rámce. Tento jazyk sa učíme počas celého štúdia na univerzite a je to náš obľúbený jazyk.

Pokiaľ ide o programovacie prostredia a použité PyCharm a GoogleCollab. PyCharm je multiplatformné integrované vývojové prostredie pre programovací jazyk Python vyvinuté spoločnosťou JetBrains na základe IntelliJ IDEA. Poskytuje používateľovi súbor nástrojov na grafické ladenie a manipuláciu s kódom. Produkt je k dispozícii v dvoch verziách: PyCharm Community Edition - bezplatná verzia, je pod licenciou Apache, a PyCharm Professional Edition - rozšírená verzia produktu s ďalšími funkciami, je to proprietárny softvér.

3 Použíte modely na rozpoznávania textu

Dlhodobá krátkodobá pamäť (**LSTM**) je umelá neurónová sieť používaná v oblasti umelej inteligencie a hlbokého učenia. Na rozdiel od štandardných dopredných neurónových sietí má LSTM spätnoväzbové spojenia. Takáto rekurentná neurónová sieť (RNN) dokáže spracovať nielen jednotlivé dátové body (napríklad obrázky), ale aj celé sekvencie dát (napríklad reč alebo video). LSTM je napríklad použiteľná na úlohy, ako je nesegmentované, prepojené rozpoznávanie rukopisu,rozpoznávanie reči, strojový preklad, riadenie robotov, videohry a zdravotníctvo. Váhy spojení a zaujatosti v sieti sa menia raz za epizódu trénovania, analogicky k tomu, ako fyziologické zmeny v synaptických silách ukladajú dlhodobé spomienky; aktivačné vzory v sieti sa menia raz za časový krok, analogicky k tomu, ako momentálna zmena elektrických vzruchov v mozgu ukladá krátkodobé spomienky. Cieľom architektúry LSTM je poskytnúť RNN krátkodobú pamäť, ktorá môže trvať tisíce časových krokov, teda "dlhodobú krátkodobú pamäť".

Gated recurrent units (**GRU**) je gatingový mechanizmus v rekurentných neurónových sieťach, ktorý v roku 2014 predstavil Kyunghyun Cho. GRU je ako dlhodobá krátkodobá pamäť (LSTM) so zabudnutým hradlom, ale má menej parametrov ako LSTM, pretože jej chýba výstupné hradlo. Zistilo sa, že výkonnosť GRU pri určitých úlohách modelovania polyfonickej hudby, modelovania rečového signálu a spracovania prirodzeného jazyka je podobná ako výkonnosť LSTM. Ukázalo

sa, že GRU vykazujú lepší výkon na určitých menších a menej častých súboroch údajov.

(BiLSTM) Pre čo najjednoduchšie vysvetlenie obojsmernej RNN si predstavte bunku RNN ako čiernu skrinku, ktorá prijíma ako vstup skrytý stav (vektor) a slovný vektor a vydáva výstupný vektor a ďalší skrytý stav. Táto skrinka má určité váhy, ktoré sa majú vyladiť pomocou spätnej propagácie strát. Tá istá bunka sa použije aj na všetky slová, takže váhy sú spoločné pre všetky slová vo vete. Toto sa nazýva zdieľanie váh. V obojsmernej RNN je jedinou zmenou to, že text čítame normálnym spôsobom aj v opačnom smere. Takže paralelne poskladáme dve RNN, a teda dostaneme 8 výstupných vektorov na pripojenie. Keď získame výstupné vektory, pošleme ich cez sériu hustých vrstiev a nakoniec cez vrstvu softmax na vytvorenie klasifikátora textu. Vo väčšine prípadov stačí pochopiť, ako ukladať niektoré vrstvy v neurónovej sieti, aby ste získali najlepšie výsledky. Možno uvažovať o tom, že v sieti bude viacero obojsmerných vrstiev GRU/LSTM, ak to bude mať lepšie výsledky.

Obojsmerný GRU alebo **BiGRU** je model spracovania sekvencie, ktorý pozostáva z dvoch GRU. jeden prijíma vstup v priamom smere a druhý v spätnom smere. Je to obojsmerná rekurentná neurónová sieť, ktorá má len vstupné a zabudovacie hradlá. V tejto štúdii sa navrhuje model BiGRU-CNN - skonštruovaný pomocou vrstvy Bi-GRU, po ktorej nasleduje vrstva CNN.

4 Programové kódy jednotlivých modelov

main.py | LSTM.py | GRU.py | BiLSTM.py | BiGRU.py

```python
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Embedding
from keras.preprocessing import sequence
from keras.datasets import imdb




#LSTM Model
def lstm_model(optymizer = "adam"):
    model = keras.models.Sequential()
    model.add(keras.layers.Embedding(10000, 128))
    model.add(keras.layers.LSTM(128, dropout=0.2, recurrent_dropout=0.2))
    model.add(keras.layers.Dense(46, activation='softmax'))
    model.compile(loss='sparse_categorical_crossentropy', optimizer=optymizer, metrics=['accuracy'])
    return model


def train_lstm_model(model, X_train, y_train, X_test, y_test,epochs=3, batch_size=64):
    model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=batch_size)
    return model

def predict_lstm_model(model, X_test, y_test):
    # Final evaluation of the model
    scores = model.evaluate(X_test, y_test, verbose=0)
    print("Accuracy: %.2f%%" % (scores[1]*100))
    return scores

```

LSTM model

main.py | LSTM.py | GRU.py | BiLSTM.py | BiGRU.py

```python
#GRU
import keras

def gru_model(optymizer = "adam"):
    model = keras.models.Sequential()
    model.add(keras.layers.Embedding(10000, 128))
    model.add(keras.layers.GRU(128, dropout=0.2, recurrent_dropout=0.2))
    model.add(keras.layers.Dense(46, activation='softmax'))
    model.compile(loss='sparse_categorical_crossentropy', optimizer=optymizer, metrics=['accuracy'])
    return model

def train_gru_model(model, X_train, y_train, X_test, y_test,epochs=3, batch_size=64):
    model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=batch_size)
    return model

def predict_gru_model(model, X_test, y_test):
    # Final evaluation of the model
    scores = model.evaluate(X_test, y_test, verbose=0)
    print("Accuracy: %.2f%%" % (scores[1]*100))
    return scores
```

GRU model

```python
import keras


def bilstm_model(optymizer ="adam"):
    model = keras.models.Sequential()
    model.add(keras.layers.Embedding(10000, 128))
    model.add(keras.layers.Bidirectional(keras.layers.LSTM(128, dropout=0.2, recurrent_dropout=0.2)))
    model.add(keras.layers.Dense(46, activation='softmax'))
    model.compile(loss='sparse_categorical_crossentropy', optimizer=optymizer, metrics=['accuracy'])
    return model

def train_bilstm_model(model, X_train, y_train, X_test, y_test,epochs=3, batch_size=64):
    model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, batch_size=batch_size)
    return model

def predict_bilstm_model(model, X_test, y_test):
    # Final evaluation of the model
    scores = model.evaluate(X_test, y_test, verbose=0)
    print("Accuracy: %.2f%%" % (scores[1]*100))
    return scores
```

Bidirectional models

```python
def preprocess_sentences(X_train, X_test):
    # Convert sentences to sequences
    tokenizer = Tokenizer(num_words=10000)
    tokenizer.fit_on_texts(X_train)
    X_train = tokenizer.texts_to_sequences(X_train)
    X_test = tokenizer.texts_to_sequences(X_test)

    # Pad sequences with zeros
    X_train = tf.keras.preprocessing.sequence.pad_sequences(X_train, maxlen=100)
    X_test = tf.keras.preprocessing.sequence.pad_sequences(X_test, maxlen=100)

    return X_train, X_test

X_train, X_test = preprocess_sentences(X_train, X_test)

for opt in optymizers:
    print("Optymizer: ", opt)
    lstm = lstm_model(opt)
    lstm = train_lstm_model(lstm, X_train, y_train, X_test, y_test)
    lstm_scores = predict_lstm_model(lstm, X_test, y_test)
    gru = gru_model(opt)
    gru = train_gru_model(gru, X_train, y_train, X_test, y_test)
    gru_scores = predict_gru_model(gru, X_test, y_test)
    bilstm = bilstm_model(opt)
    bilstm = train_bilstm_model(bilstm, X_train, y_train, X_test, y_test)
    bilstm_scores = predict_bilstm_model(bilstm, X_test, y_test)
    bigru = bigru_model(opt)
    bigru = train_bigru_model(bigru, X_train, y_train, X_test, y_test)
    bigru_scores = predict_bigru_model(bigru, X_test, y_test)
    print("LSTM: %.2f%%" % (lstm_scores[1]*100))
    print("GRU: %.2f%%" % (gru_scores[1]*100))
    print("BiLSTM: %.2f%%" % (bilstm_scores[1]*100))
    print("BiGRU: %.2f%%" % (bigru_scores[1]*100))
    print("_"*50)
```

Main code

5 Výsledky a úspešnosť vyškolených modelov

Tu uvidíte, ako úspešne naše vyškolené modely dokázali odhaliť trollingový text a urážky, výsledky sú potešiteľné.

```
Epoch 1/3
183/183 [==============================] - 50s 256ms/step - loss: 0.7213 - accuracy: 0.7087 - val_loss: 0.2353 - val_accuracy: 0.9110
Epoch 2/3
183/183 [==============================] - 43s 236ms/step - loss: 0.1511 - accuracy: 0.9473 - val_loss: 0.1162 - val_accuracy: 0.9607
Epoch 3/3
183/183 [==============================] - 43s 235ms/step - loss: 0.0595 - accuracy: 0.9823 - val_loss: 0.1461 - val_accuracy: 0.9553
Accuracy: 95.53%
```

Výsledky modelu LSTM ukázali úspešnosť približne 95.5 %.

```
Epoch 1/3
183/183 [==============================] - 38s 196ms/step - loss: 0.8410 - accuracy: 0.6669 - val_loss: 0.3562 - val_accuracy: 0.8569
Epoch 2/3
183/183 [==============================] - 35s 193ms/step - loss: 0.1578 - accuracy: 0.9425 - val_loss: 0.1121 - val_accuracy: 0.9595
Epoch 3/3
183/183 [==============================] - 36s 196ms/step - loss: 0.0426 - accuracy: 0.9872 - val_loss: 0.0922 - val_accuracy: 0.9719
Accuracy: 97.19%
```

Výsledky modelu GRU ukázali úspešnosť nad 97 %.

```
Epoch 1/3
183/183 [==============================] - 120s 625ms/step - loss: 0.6099 - accuracy: 0.7577 - val_loss: 0.1895 - val_accuracy: 0.9251
Epoch 2/3
183/183 [==============================] - 151s 824ms/step - loss: 0.1186 - accuracy: 0.9588 - val_loss: 0.1196 - val_accuracy: 0.9615
Epoch 3/3
183/183 [==============================] - 157s 860ms/step - loss: 0.0485 - accuracy: 0.9862 - val_loss: 0.0630 - val_accuracy: 0.9818
Accuracy: 98.18%
```

Výsledky modelu BiLSTM ukázali úspešnosť 98.2 %.

```
Epoch 1/3
183/183 [==============================] - 125s 653ms/step - loss: 0.6383 - accuracy: 0.7702 - val_loss: 0.1583 - val_accuracy: 0.9405
Epoch 2/3
183/183 [==============================] - 120s 657ms/step - loss: 0.0844 - accuracy: 0.9734 - val_loss: 0.0739 - val_accuracy: 0.9728
Epoch 3/3
183/183 [==============================] - 126s 684ms/step - loss: 0.0336 - accuracy: 0.9900 - val_loss: 0.0702 - val_accuracy: 0.9780
Accuracy: 97.80%
```

Výsledky modelu BiGRU ukázali úspešnosť 97.8 %.

```
LSTM: 95.53%
GRU: 97.19%
BiLSTM: 98.18%
BiGRU: 97.80%
```

Každý z modelov bol tiež testovaný s rôznymi optimalizátormi, ako napr. **['adam', 'rmsprop', 'adagrad', 'adadelta', 'adamax', 'nadam']** výsledky však kolísali veľmi mieme, približne +-2 %.

Závery

Cieľom tejto práce bolo vytvoriť model neurónovej siete, ktorý dokáže klasifikovať toxický obsah online diskusií s dostatočnou presnosťou. Ako už bolo uvedené, náš model hlbokej neurónovej siete BiLSTM dosiahol najlepší výkon Acc=0,982 pomocou optimalizátora Adam. Najlepším optimalizátorom bol Adam pre všetky typy sietí (LSTM, GRU a BiLSTM, BiGRU). Najlepší hlboký model možno úspešne použiť na rozpoznávanie útočného obsahu.

Môže byť to použite na vytvorenie webovej služby, ktorá poskytuje používateľom sociálnej platformy podrobné informácie o danom komentári ako jeden výber z možností: komentár je toxický, alebo komentár nie je toxický.

Do budúcnosti by sme chceli rozpoznať urážlivé prejavy z rôznych typov sledovaných údajov. Chceli by sme uvažovať o obrazových údajoch, ktoré by dopĺňali text komentárov na rozlíšenie toxických emócií.

Zaujímavým výskumom do budúcnosti by mohlo byť preskúmanie možnosti zvýšenia efektivity nášho modelu na vytvorenie novych vrstv modelu.

Bolo pre nás celkom zaujímavé pracovať na tejto oblasti, ktorá teraz pokrýva takmer všetky oblasti činnosti. Naše výsledky sú pomerne vysoké a možno ich porovnať s oveľa väčšími prácami s profesionálnejším prístupom.