Лабораторна робота № 11. Моделювання нейронної мережі. Класифікація текстів з використанням RNN

Мета: сформулювати практичні навички у розробці, тренуванні та оцінці нейронних мереж з використанням рекурентних нейронних мереж (RNN) для задач класифікації текстів. Розвинути уміння працювати з текстовими даними, виконувати їх попередню обробку та векторизацію, а також оцінювати ефективність моделей на практичних прикладах.

Теоретичні відомості

Рекурентні нейронні мережі (RNN) є класом нейронних мереж, що ефективно обробляють послідовні дані або часові ряди. Вони здатні зберігати інформацію про попередні елементи послідовності в своїй внутрішній пам’яті, що робить їх ідеальними для задач обробки природної мови, включаючи класифікацію текстів, генерацію тексту, переклад мови та інші.

Основні компоненти RNN включають вузли (нейрони), які пов’язані між собою зворотними зв’язками, дозволяючи інформації циркулювати всередині мережі. Ця особливість дозволяє RNN «пам’ятати» попередні входи та використовувати цю інформацію для вирахування виходу.

Проте, стандартні RNN стикаються з проблемою зникнення або вибуху градієнтів при тренуванні на довгих послідовностях. Цю проблему частково вирішують варіанти RNN, такі як **LSTM (Long Short-Term Memory)** та **GRU (Gated Recurrent Unit)**, які впроваджують механізми забування та оновлення інформації, щоб балансувати збереження та відкидання інформації через час.

Попередня обробка тексту є критичним кроком при роботі з задачами класифікації текстів. Це включає видалення шуму (наприклад, пунктуації та стоп-слів), токенізацію тексту, векторизацію слів або використання предтренованих векторних представлень слів (наприклад, Word2Vec або GloVe), та нормалізацію даних перед подачею їх у модель.

Для оцінки ефективності моделей класифікації текстів використовуються такі метрики, як точність (accuracy), точність класифікації для кожного класу (precision), повнота (recall), F1-оцінка, яка є гармонійним середнім між точністю та повнотою, а також матриця помилок для візуалізації помилок класифікації.

Приклад виконання роботи

Завдання

1. **Підготовка даних.**
   - Завантажте набір даних для класифікації текстів, який містить текстові зразки та їх відповідні мітки класів (наприклад, позитивний, негативний, нейтральний);
   - Виконайте попередню обробку текстових даних: очищення тексту, видалення зайвих символів, токенізація, конвертація тексту в послідовності та їх подальше вирівнювання (padding).
2. **Розробка моделі RNN.**
   - Створіть модель RNN з використанням шарів Embedding, LSTM або GRU (за вибором) для обробки послідовностей;
   - Додайте необхідні шари для класифікації, такі як Dense, Dropout для запобігання перенавчанню;
   - Компілюйте модель, вибравши відповідну функцію втрат та оптимізатор.
3. **Тренування та оцінка моделі.**
   - Тренуйте модель на тренувальному наборі даних, використовуючи валідаційний набір для моніторингу процесу навчання;
   - Використайте колбеки, такі як EarlyStopping та ReduceLROnPlateau, для оптимізації процесу тренування;
   - Оцініть ефективність моделі на тестовому наборі даних, аналізуючи метрики, такі як точність (accuracy).
4. **Аналіз результатів.**
   - Візуалізуйте історію тренувань за допомогою графіків для оцінки змін точності та втрат в процесі тренування;
   - Проведіть декілька тестів з власними текстами для перевірки здатності моделі правильно класифікувати текстові дані.
5. **Експериментування.**
   - Модифікуйте архітектуру моделі, кількість епох, розмір пакету (batch size), оптимізатор та інші параметри тренування для покращення результатів;
   - Спробуйте використати різні стратегії попередньої обробки даних та інженерії

ознак для аналізу їх впливу на ефективність моделі.

Контрольне запитання №1

Вкажіть результуюче значення Accuracy.
Відповідь: _____

Контрольне запитання №2

Яка основна проблема стандартних RNN при тренуванні на довгих послідовностях?
Відповідь: _____

Контрольне запитання №3

Які дві модифікації RNN розроблені для вирішення цієї проблеми?
Відповідь: _____

Хід роботи

1. Збережемо та імпортуємо набори даних.

textID	text	selected_text	sentiment	Time of Tweet	Age of User	Country	Population -2020	Land Area (Km◆)	Density (P/Km◆)
cb774db0d1	I'd have responded,	I'd have responded,	neutral	morning	0-20	Afghanistan	38928346	652860	60
549e992a42	Sooo SAD I will miss	Sooo SAD	negative	noon	21-30	Albania	2877797	27400	105
088c60f138	my boss is bullying m	bullying me	negative	night	31-45	Algeria	43851044	2381740	18
9642c003ef	what interview! leave	leave me alone	negative	morning	46-60	Andorra	77265	470	164
358bd9e861	Sons of ****, why co	Sons of ****,	negative	noon	60-70	Angola	32866272	1246700	26
28b57f3990	http://www.dothebou	http://www.dothebou	neutral	night	70-100	Antigua and Barbud	97929	440	223

Рисунок 10.1. Датасет на Google Drive

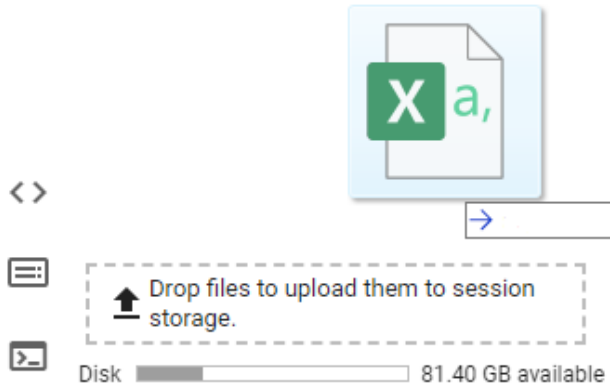

Рисунок 10.2. Завантаження датасету до Google Colab

2. Встановлення необхідних бібліотек та модулів.

```
import warnings
warnings.filterwarnings('ignore')
```

```
from keras.preprocessing.text import Tokenizer
from keras.utils import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, Dropout, Bidirectional
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
from keras.utils import to_categorical
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

3. Завантаження наборів даних.

```
train_ds = pd.read_csv('/content/train.csv',encoding='latin1');
validation_ds = pd.read_csv('/content/test.csv',encoding='latin1');

train_ds.head()
validation_ds.head()
```

	textID	text	selected_text	sentiment	Time of Tweet	Age of User	Country	Population -2020	Land Area (Km²)	Density (P/Km²)
0	cb774db0d1	I`d have responded, if I were going	I`d have responded, if I were going	neutral	morning	0-20	Afghanistan	38928346	652860.0	60
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative	noon	21-30	Albania	2877797	27400.0	105
2	088c60f138	my boss is bullying me...	bullying me	negative	night	31-45	Algeria	43851044	2381740.0	18
3	9642c003ef	what interview! leave me alone	leave me alone	negative	morning	46-60	Andorra	77265	470.0	164
4	358bd9e861	Sons of ****, why couldn`t they put them on t...	Sons of ****,	negative	noon	60-70	Angola	32866272	1246700.0	26

Рисунок 10.3. Перші 5 рядків навчального набору даних

	textID	text	sentiment	Time of Tweet	Age of User	Country	Population -2020	Land Area (Km²)	Density (P/Km²)
0	f87dea47db	Last session of the day http://twitpic.com/67ezh	neutral	morning	0-20	Afghanistan	38928346.0	652860.0	60.0
1	96d74cb729	Shanghai is also really exciting (precisely -...	positive	noon	21-30	Albania	2877797.0	27400.0	105.0
2	eee518ae67	Recession hit Veronique Branquinho, she has to...	negative	night	31-45	Algeria	43851044.0	2381740.0	18.0
3	01082688c6	happy bday!	positive	morning	46-60	Andorra	77265.0	470.0	164.0
4	33987a8ee5	http://twitpic.com/4w75p - I like it!!	positive	noon	60-70	Angola	32866272.0	1246700.0	26.0

Рисунок 10.4. Перші 5 рядків тестового набору даних

Ми бачимо, що загальна кількість стовпців у тестовому наборі даних на один менше, ніж у навчальному. Відсутній стовпець у тестових даних – це selected_text (виділений текст) стовпець, який ми повинні передбачити.

```
print("Загальна кількість рядків у навчальному наборі даних ",
train_ds.shape[0])
print("Загальна кількість стовпців у навчальному наборі даних ",
train_ds.shape[1])
print("Загальна кількість рядків у тестовому наборі даних ",
validation_ds.shape[0])
print("Загальна кількість стовпців у тестовому наборі даних ",
validation_ds.shape[1])
```

```
Загальна кількість рядків у навчальному наборі даних  27481
Загальна кількість стовпців у навчальному наборі даних  10
Загальна кількість рядків у тестовому наборі даних  4815
Загальна кількість стовпців у тестовому наборі даних  9
```

Рисунок 10.5. Загальна кількість рядків та стовпців у тестовому та навчальному наборах даних

4. Візуалізація та аналіз даних

Виконаємо візуалізацію кількості нульових значень в обох наборах даних

```
plt.figure(figsize = (13,5))
plt.bar(train_ds.columns, train_ds.isna().sum())
plt.xlabel("Назва стовпця")
plt.ylabel("Кількість NULL значень у навчальному наборі даних")
plt.show()
```

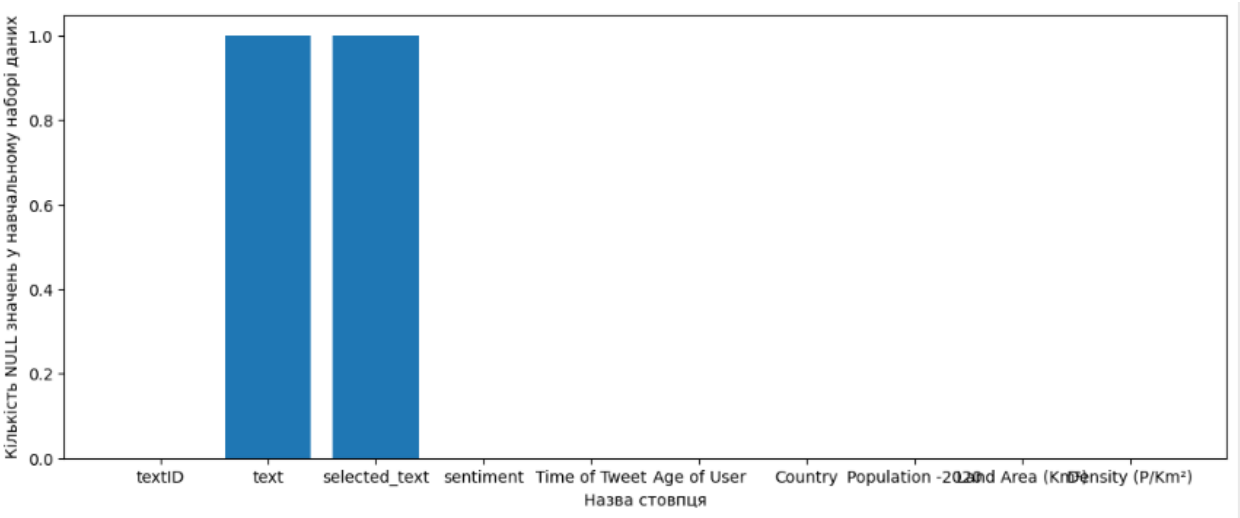

Рисунок 10.6. Графік «Кількість NULL значень у навчальному наборі даних»

```
plt.figure(figsize = (13,5))
plt.bar(validation_ds.columns, validation_ds.isnull().sum().values, color
= 'red')
plt.xlabel("Назва стовпця")
plt.ylabel("Кількість NULL значень у тестовому наборі даних")
plt.show()
```

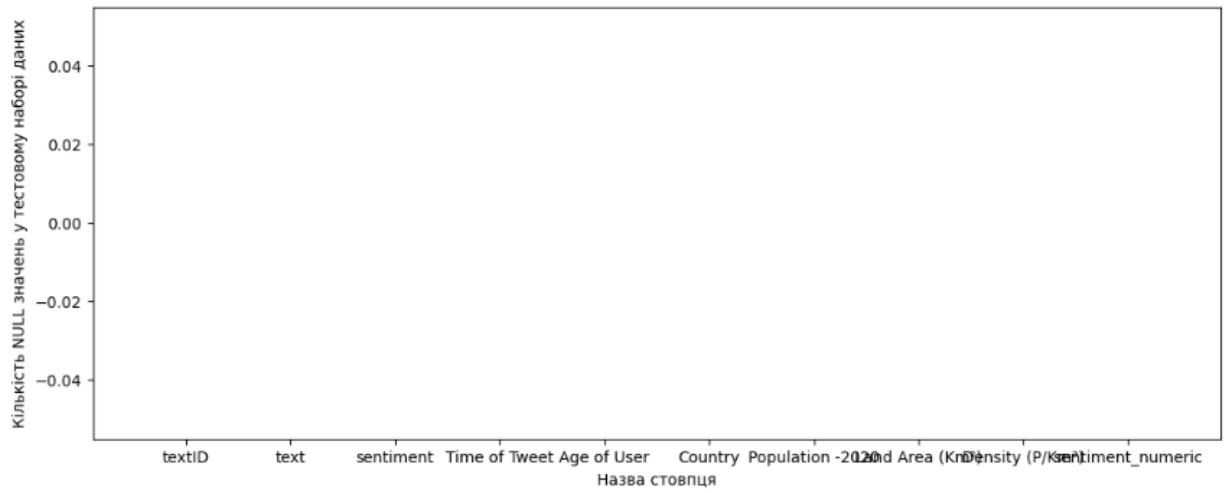

Рисунок 10.7. Графік «Кількість NULL значень у тестовому наборі даних»

Визначимо функцію для перетворення текстових міток настроїв («позитивний», «негативний», «нейтральний») на числові значення (0, 1, 2). Застосуємо цю функцію до навчального та валідаційного наборів даних для створення нових стовпчиків з числовими значеннями настроїв. Рядки з невідображеними настроями (позначені -1) відфільтровуються.

```
def sentiment_to_numeric(sentiment):
    mapping = {'positive': 0, 'negative': 1, 'neutral': 2}
    return mapping.get(sentiment, -1)

train_ds['sentiment_numeric'] =
train_ds['sentiment'].apply(sentiment_to_numeric)
validation_ds['sentiment_numeric'] =
validation_ds['sentiment'].apply(sentiment_to_numeric)

train_ds = train_ds[train_ds['sentiment_numeric'] != -1]
validation_ds = validation_ds[validation_ds['sentiment_numeric'] != -1]
```

Визначимо масив ознак (текст) і цільовий масив (настрої).

```
x_train, y_train = np.array(train_ds['text'].tolist()),
np.array(train_ds['sentiment'].tolist())
x_test, y_test = np.array(validation_ds['text'].tolist()),
np.array(validation_ds['sentiment'].tolist())
```

Перетворимо числові мітки настроїв у формат однократного кодування.

```
y_train = np.array(train_ds['sentiment_numeric'].tolist())
y_test = np.array(validation_ds['sentiment_numeric'].tolist())

y_train = to_categorical(y_train, 3)
y_test = to_categorical(y_test, 3)
```

Переглянемо розподіл категорій настроїв.

```
print("Unique values in y_train:", np.unique(y_train))
print("Unique values in y_test:", np.unique(y_test))
```

```
Unique values in y_train: [0. 1.]
Unique values in y_test: [0. 1.]
```

Рисунок 10.8. Унікальні значення категорій настроїв у навчальному та тестовому наборах

Підготуємо текстові дані для введення в модель шляхом токенізації та розбиття на частини.

```
# Tokenization and padding
tokenizer = Tokenizer(num_words=20000)
tokenizer.fit_on_texts(list(x_train) + list(x_test))
x_train_seq = tokenizer.texts_to_sequences(x_train)
x_test_seq = tokenizer.texts_to_sequences(x_test)
x_train_pad = pad_sequences(x_train_seq, maxlen=35)
x_test_pad = pad_sequences(x_test_seq, maxlen=35)
```

5. Розробимо та охарактеризуємо модель нейронної мережі

Змоделюємо послідовну архітектуру моделі з вбудовуванням, двонаправленими LSTM шарами, відсівом для регуляризації та щільним шаром для класифікації.

Примітка. Під час розробки ви можете використовувати різну кількість нейронів для кожного шару, різне значення для відсіву. Ви також можете змінювати значення гіперпараметрів для отримання кращого результату.

```
# Нейронна мережа
model = Sequential([
    Embedding(input_dim=20000, output_dim=100, input_length=35),
    Bidirectional(LSTM(64, return_sequences=True)),
    Dropout(0.5),
    Bidirectional(LSTM(32)),
    Dense(3, activation='softmax')
])
```

Підготуємо модель до навчання, задаючи оптимізатор, функцію втрат та метрики.

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

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 35, 100)           2000000

 bidirectional (Bidirection  (None, 35, 128)           84480
 al)

 dropout (Dropout)           (None, 35, 128)           0

 bidirectional_1 (Bidirecti  (None, 64)                41216
 onal)

 dense (Dense)               (None, 3)                 195

=================================================================
Total params: 2125891 (8.11 MB)
Trainable params: 2125891 (8.11 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Рисунок 10.9. Характеристика моделі нейронної мережі

Налаштуємо зворотні дзвінки, щоб запобігти надмірному пристосуванню.

```
# Callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=3)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2,
min_lr=0.001)
```

Виконаємо процес навчання та оцінювання роботи моделі.

```
# Model training
history = model.fit(x_train_pad, y_train, epochs=5,
validation_data=(x_test_pad, y_test), callbacks=[early_stopping,
reduce_lr])
```

```
Epoch 1/5
859/859 [==============================] - 48s 40ms/step - loss: 0.7694 - accuracy: 0.6497 - val_loss: 0.6524
Epoch 2/5
859/859 [==============================] - 14s 16ms/step - loss: 0.5275 - accuracy: 0.7894 - val_loss: 0.6543
Epoch 3/5
859/859 [==============================] - 13s 15ms/step - loss: 0.3918 - accuracy: 0.8504 - val_loss: 0.7467
Epoch 4/5
859/859 [==============================] - 12s 14ms/step - loss: 0.2951 - accuracy: 0.8926 - val_loss: 0.8033
```

Рисунок 10.10. Процес тренування RNN моделі

6. Візуалізуємо точність навчання та валідації моделі.

```
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.legend()
plt.show()
```

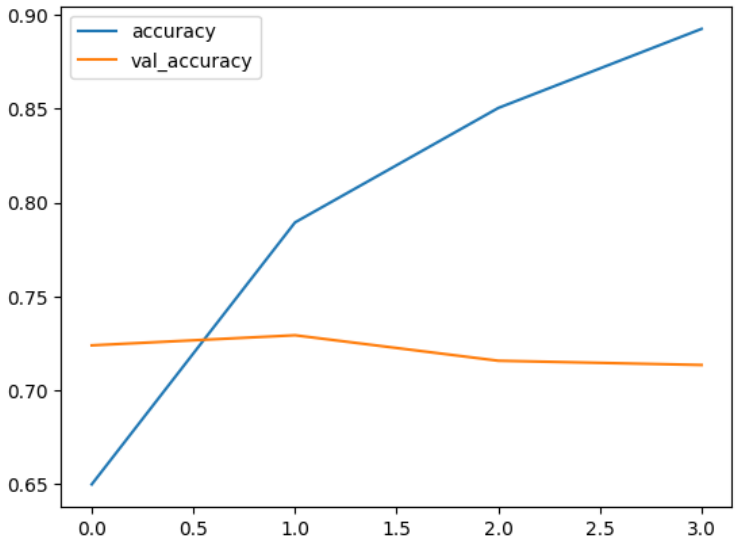

Рисунок 10.11. Графік точності навчання

7. Створимо функцію передбачення настрою в текстовому контексті.

Визначимо функцію для попередньої обробки довільного тексту (токенізація та паддінг), передбачення настрою за допомогою навченої моделі та декодування передбачення у читабельну мітку настрою.

```
def predict_sentiment(text, tokenizer, model):
    seq = tokenizer.texts_to_sequences([text])
    padded_seq = pad_sequences(seq, maxlen=35)
    prediction = model.predict(padded_seq)
    sentiment_idx = np.argmax(prediction)
    sentiments = {0: 'Positive', 1: 'Negative', 2: 'Neutral'}
    return sentiments[sentiment_idx]

# Example 1:
text_to_predict = "I love this product, it's amazing!"
predicted_sentiment = predict_sentiment(text_to_predict, tokenizer, model)
print(f"Predicted sentiment: {predicted_sentiment}")

# Example 2:
text_to_predict = "This is the worst movie I have ever seen."
predicted_sentiment = predict_sentiment(text_to_predict, tokenizer, model)
print(f"Predicted sentiment: {predicted_sentiment}")
```

```
1/1 [==============================] - 2s 2s/step
Predicted sentiment: Positive
1/1 [==============================] - 0s 24ms/step
Predicted sentiment: Negative
```

Рисунок 9.12. Приклад передбачення моделлю RNN

Контрольне запитання №1

Вкажіть результуюче значення Accuracy.
Відповідь: **0.8926**

Контрольне запитання №2

Яка основна проблема стандартних RNN при тренуванні на довгих послідовностях?
Відповідь: _____

Контрольне запитання №3

Які дві модифікації RNN розроблені для вирішення цієї проблеми?
Відповідь: _____

Висновки: …

Завдання для виконання лабораторної роботи

1. **Підготовка даних.**
   - Завантажте набір даних для класифікації текстів, який містить текстові зразки та їх відповідні мітки класів (наприклад, позитивний, негативний, нейтральний);
   - Виконайте попередню обробку текстових даних: очищення тексту, видалення зайвих символів, токенізація, конвертація тексту в послідовності та їх подальше вирівнювання (padding).
2. **Розробка моделі RNN.**
   - Створіть модель RNN з використанням шарів Embedding, LSTM або GRU (за вибором) для обробки послідовностей;
   - Додайте необхідні шари для класифікації, такі як Dense, Dropout для запобігання перенавчання;
   - Компілюйте модель, вибравши відповідну функцію втрат та оптимізатор.
3. **Тренування та оцінка моделі.**
   - Тренуйте модель на тренувальному наборі даних, використовуючи валідаційний набір для моніторингу процесу навчання;
   - Використайте колбеки, такі як EarlyStopping та ReduceLROnPlateau, для оптимізації процесу тренування;
   - Оцініть ефективність моделі на тестовому наборі даних, аналізуючи метрики, такі як точність (accuracy).
4. **Аналіз результатів.**
   - Візуалізуйте історію тренувань за допомогою графіків для оцінки змін точності та втрат в процесі тренування;
   - Проведіть декілька тестів з власними текстами для перевірки здатності моделі правильно класифікувати текстові дані.
5. **Експериментування.**
   - Модифікуйте архітектуру моделі, кількість епох, розмір пакету (batch size), оптимізатор та інші параметри тренування для покращення результатів;
   - Спробуйте використати різні стратегії попередньої обробки даних та інженерії ознак для аналізу їх впливу на ефективність моделі.

Примітка. Дана лабораторна робота не має конкретних варіантів, але передбачає, що виконання буде базуватися на принципах креативності та зацікавленості здобувачів.

Посилання на навчальний та тестовий набори даних

Контрольне запитання №1

Вкажіть результуюче значення Accuracy.
Відповідь: _____

Контрольне запитання №2

Яка основна проблема стандартних RNN при тренуванні на довгих послідовностях?
Відповідь: _____

Контрольне запитання №3

Які дві модифікації RNN розроблені для вирішення цієї проблеми?
Відповідь: _____

Welcome to Google Colab

https://colab.research.google.com/drive/1vowoil7SsUzJ6Vx3p-xSldiJBkv8jepp?usp=sharing