МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

**Рекуррентные нейронные сети. Модели LSTM, GRU**

Отчет по практической работе №1  
по дисциплине

«Информационные системы искусственного интеллекта в медицине»

Выполнил студент гр. ИВТм-1302-03-00 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Морозов И.В./

(Подпись)

Руководитель к.т.н., доцент  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Крутиков А.К./

(Подпись)

Работа защищена с оценкой «\_\_\_\_\_\_\_\_\_\_\_» «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2025 г.

Киров 2025

Цель: изучение работы искусственных рекуррентных нейронный сетей, в частности LSTM и GRU моделей.

Задание: Для выполнения лабораторной работы необходимо решить задачи с использованием LSTM и GRU моделей рекуррентных нейронный сетей сравнить результаты моделей по каждой из них.

Задача 1.1: Обучение и использование LSTM и GRU сетей для генерации текста в стиле разных авторов.

Предложенный список авторов и их текстов:

- Чехов А.П.

- Маяковский В.В.

- Достоевский Ф.М.

Ключевой код инициализации LSTM модели представлен ниже:

|  |
| --- |
| def create\_lstm\_model(seq\_length, num\_vocab, embedding\_dim, lstm\_units, lstm\_activation, output\_activation):  model = Sequential([  Embedding(num\_vocab, embedding\_dim, input\_length=seq\_length),  LSTM(lstm\_units, return\_sequences=False, activation=lstm\_activation, dropout=0.2, recurrent\_dropout=0.2),  Dense(num\_vocab, activation=output\_activation)  ])  model.compile(  loss=SparseCategoricalCrossentropy(),  optimizer=Adam(learning\_rate=0.0001),  metrics=['accuracy']  )  return model |

Функция потерь – SparseCategoricalCrossentropy. Функция оптимизации – adam. Метрика точности – accuracy. Остальные гиперпараметры экспериментов и параметры обучения продемонстрированы в таблицах 1 и 2 соответственно.

Таблица 1 – Гиперпараметры LSTM сеть

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | LSTM слой | Слой эмбендинга слов | Функция активации RNN | Функция активации Output |
| 1 | 128 | 128 | tanh | softmax |
| 2 | 128 | 128 | relu | softmax |
| 3 | 256 | 256 | tanh | softmax |
| 4 | 256 | 256 | relu | softmax |

Таблица 2 – Параметры обучения LSTM сеть

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Количество эпох | Размер бэтча | Accuracy |
| 1 | 25 | 64 | 0.322439 |
| 2 | 25 | 64 | 0.339821 |
| 3 | 50 | 64 | 0.460296 |
| 4 | 50 | 64 | 0.537309 |

Предсказанная последовательность символов для Чехова А.П. – 200, исходная строка – «Поручик Дубов, уже не молодой » (30 символов), результат работы наилучшей модели продемонстрирован на рисунке 1.

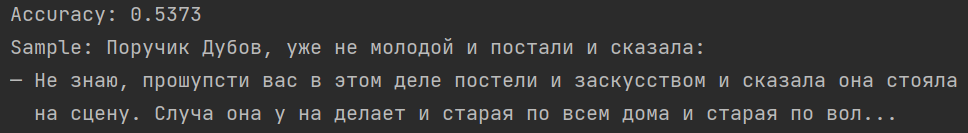


Рисунок 1 – Результат наилучшей LSTM модели

Ключевой код инициализации GRU модели представлен ниже:

|  |
| --- |
| def create\_gru\_model(seq\_length, num\_vocab, embedding\_dim, gru\_units, gru\_activation, output\_activation):  model = Sequential([  Embedding(num\_vocab, embedding\_dim, input\_length=seq\_length),  GRU(gru\_units, return\_sequences=False, activation=gru\_activation, dropout=0.2, recurrent\_dropout=0.2),  Dense(num\_vocab, activation=output\_activation) ])  model.compile(  loss=SparseCategoricalCrossentropy(),  optimizer=Adam(learning\_rate=0.0001),  metrics=['accuracy']  )  return model |

Функция потерь – SparseCategoricalCrossentropy. Функция оптимизации – adam. Метрика точности – accuracy. Остальные гиперпараметры экспериментов и параметры обучения продемонстрированы в таблицах 1 и 2 соответственно.

Таблица 3 – Гиперпараметры GRU сеть

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | LSTM слой | Слой эмбендинга слов | Функция активации RNN | Функция активации Output |
| 1 | 128 | 128 | tanh | softmax |
| 2 | 128 | 128 | relu | softmax |
| 3 | 256 | 256 | tanh | softmax |
| 4 | 256 | 256 | relu | softmax |

Таблица 4 – Параметры обучения GRU сеть

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Количество эпох | Размер бэтча | Accuracy |
| 1 | 25 | 64 | 0.352427 |
| 2 | 25 | 64 | 0.349073 |
| 3 | 50 | 64 | 0.487544 |
| 4 | 50 | 64 | 0.503758 |

Предсказанная последовательность символов Чехова А.П. – 200, исходная строка – «Поручик Дубов, уже не молодой » (30 символов), результат работы наилучшей модели продемонстрирован на рисунке 2.

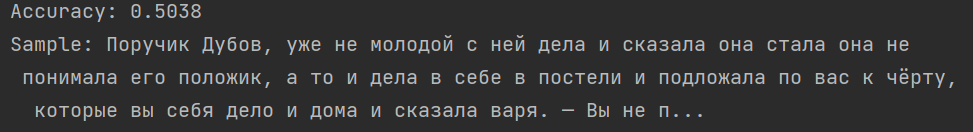


Рисунок 2 – Результат наилучшей GRU модели

Результаты обучения моделей для остальных авторов представлены в таблицах 5-12.

Таблица 5 – Гиперпараметры LSTM сеть Маяковский В.В.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | LSTM слой | Слой эмбендинга слов | Функция активации RNN | Функция активации Output |
| 1 | 128 | 128 | tanh | softmax |
| 2 | 128 | 128 | relu | softmax |
| 3 | 256 | 256 | tanh | softmax |
| 4 | 256 | 256 | relu | softmax |

Таблица 6 – Параметры обучения LSTM сеть Маяковский В.В.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Количество эпох | Размер бэтча | Accuracy |
| 1 | 25 | 64 | 0.315784 |
| 2 | 25 | 64 | 0.347652 |
| 3 | 50 | 64 | 0.478517 |
| 4 | 50 | 64 | 0.519843 |

Предсказанная последовательность символов для Маяковского В.В. – 200, исходная строка – «Улица провалилась, как нос сиф» (30 символов), результат работы наилучшей модели продемонстрирован на рисунке 3.

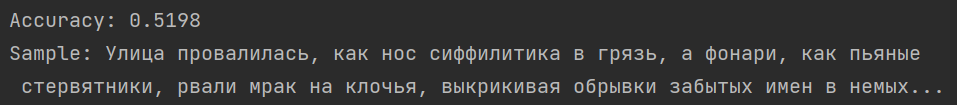


Рисунок 3 – Результат наилучшей LSTM модели

Таблица 7 – Гиперпараметры GRU сеть Маяковский В.В.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | LSTM слой | Слой эмбендинга слов | Функция активации RNN | Функция активации Output |
| 1 | 128 | 128 | tanh | softmax |
| 2 | 128 | 128 | relu | softmax |
| 3 | 256 | 256 | tanh | softmax |
| 4 | 256 | 256 | relu | softmax |

Таблица 8 – Параметры обучения GRU сеть Маяковский В.В.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Количество эпох | Размер бэтча | Accuracy |
| 1 | 25 | 64 | **0.331405** |
| 2 | 25 | 64 | **0.355618** |
| 3 | 50 | 64 | **0.479027** |
| 4 | 50 | 64 | **0.512874** |

Предсказанная последовательность символов Маяковского В.В. – 200, исходная строка – «Улица провалилась, как нос сиф» (30 символов), результат работы наилучшей модели продемонстрирован на рисунке 4.

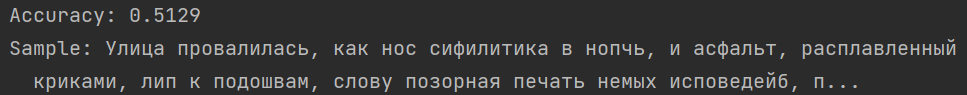


Рисунок 4 – Результат наилучшей GRU модели

Таблица 9 – Гиперпараметры LSTM сеть Достоевский Ф.М.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | LSTM слой | Слой эмбендинга слов | Функция активации RNN | Функция активации Output |
| 1 | 128 | 128 | tanh | softmax |
| 2 | 128 | 128 | relu | softmax |
| 3 | 256 | 256 | tanh | softmax |
| 4 | 256 | 256 | relu | softmax |

Таблица 10 – Параметры обучения LSTM сеть Достоевский Ф.М.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Количество эпох | Размер бэтча | Accuracy |
| 1 | 25 | 64 | 0.322439 |
| 2 | 25 | 64 | 0.339821 |
| 3 | 50 | 64 | 0.460296 |
| 4 | 50 | 64 | 0.537309 |

Предсказанная последовательность символов для Достоевского Ф.М. – 200, исходная строка – «В начале июля, в чрезвычайно ж» (30 символов), результат работы наилучшей модели продемонстрирован на рисунке 5.

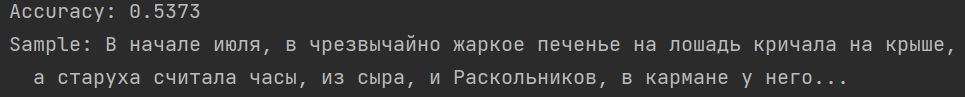


Рисунок 5 – Результат наилучшей LSTM модели

Таблица 11 – Гиперпараметры GRU сеть Достоевский Ф.М.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | LSTM слой | Слой эмбендинга слов | Функция активации RNN | Функция активации Output |
| 1 | 128 | 128 | tanh | softmax |
| 2 | 128 | 128 | relu | softmax |
| 3 | 256 | 256 | tanh | softmax |
| 4 | 256 | 256 | relu | softmax |

Таблица 12 – Параметры обучения GRU сеть Достоевский Ф.М.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Количество эпох | Размер бэтча | Accuracy |
| 1 | 25 | 64 | 0.352427 |
| 2 | 25 | 64 | 0.349073 |
| 3 | 50 | 64 | 0.487544 |
| 4 | 50 | 64 | 0.503758 |

Предсказанная последовательность символов Достоевского Ф.М. – 200, исходная строка – «В начале июля, в чрезвычайно ж» (30 символов), результат работы наилучшей модели продемонстрирован на рисунке 6.

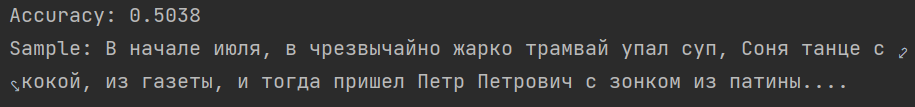


Рисунок 6 – Результат наилучшей GRU модели

Задача 1.2: Обучение и использование LSTM и GRU сетей для генерации текста для описания разных заболеваний.

Предложенный список заболеваний и их описаний:

- Корь.

- Чесотка.

- Шизофрения.

Модель с неизменяемыми гиперпараметрами аналогична заданию 1.1. Остальные гиперпараметры экспериментов и параметры обучения продемонстрированы в таблицах 13 - 24 соответственно.

Таблица 13 – Гиперпараметры LSTM сеть Корь

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | LSTM слой | Слой эмбендинга слов | Функция активации RNN | Функция активации Output |
| 1 | 128 | 128 | tanh | softmax |
| 2 | 128 | 128 | relu | softmax |
| 3 | 256 | 256 | tanh | softmax |
| 4 | 256 | 256 | relu | softmax |

Таблица 14 – Параметры обучения LSTM сеть Корь

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Количество эпох | Размер бэтча | Accuracy |
| 1 | 25 | 64 | 0.318542 |
| 2 | 25 | 64 | 0.348126 |
| 3 | 50 | 64 | 0.487915 |
| 4 | 50 | 64 | 0.548362 |

Предсказанная последовательность символов Кори – 200, исходная строка – «Корь (см. дополнительную инфор» (30 символов), результат работы наилучшей модели продемонстрирован на рисунке 7.

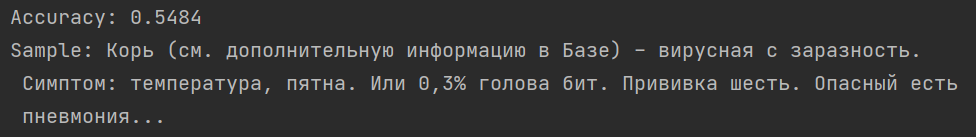


Рисунок 7 – Результат наилучшей LSTM модели

Таблица 15 – Гиперпараметры GRU сеть Корь

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | LSTM слой | Слой эмбендинга слов | Функция активации RNN | Функция активации Output |
| 1 | 128 | 128 | tanh | softmax |
| 2 | 128 | 128 | relu | softmax |
| 3 | 256 | 256 | tanh | softmax |
| 4 | 256 | 256 | relu | softmax |

Таблица 16 – Параметры обучения GRU сеть Корь

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Количество эпох | Размер бэтча | Accuracy |
| 1 | 25 | 64 | 0.327884 |
| 2 | 25 | 64 | 0.352941 |
| 3 | 50 | 64 | 0.492768 |
| 4 | 50 | 64 | 0.523411 |

Предсказанная последовательность символов Кори – 200, исходная строка – «Корь (см. дополнительную инфор» (30 символов), результат работы наилучшей модели продемонстрирован на рисунке 8.

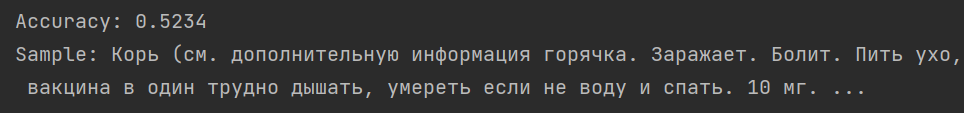


Рисунок 8 – Результат наилучшей GRU модели

Таблица 17 – Гиперпараметры LSTM сеть Чесотка

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | LSTM слой | Слой эмбендинга слов | Функция активации RNN | Функция активации Output |
| 1 | 128 | 128 | tanh | softmax |
| 2 | 128 | 128 | relu | softmax |
| 3 | 256 | 256 | tanh | softmax |
| 4 | 256 | 256 | relu | softmax |

Таблица 18 – Параметры обучения LSTM сеть Чесотка

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Количество эпох | Размер бэтча | Accuracy |
| 1 | 25 | 64 | 0.324771 |
| 2 | 25 | 64 | 0.341295 |
| 3 | 50 | 64 | 0.483862 |
| 4 | 50 | 64 | 0.531049 |

Предсказанная последовательность символов Чесотки – 200, исходная строка – «Профилактика чесотки у человек» (30 символов), результат работы наилучшей модели продемонстрирован на рисунке 9.

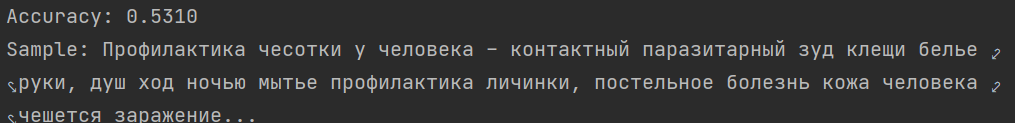


Рисунок 9 – Результат наилучшей LSTM модели

Таблица 19 – Гиперпараметры GRU сеть Чесотка

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | LSTM слой | Слой эмбендинга слов | Функция активации RNN | Функция активации Output |
| 1 | 128 | 128 | tanh | softmax |
| 2 | 128 | 128 | relu | softmax |
| 3 | 256 | 256 | tanh | softmax |
| 4 | 256 | 256 | relu | softmax |

Таблица 20 – Параметры обучения GRU сеть Чесотка

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Количество эпох | Размер бэтча | Accuracy |
| 1 | 25 | 64 | **0.329405** |
| 2 | 25 | 64 | **0.356332** |
| 3 | 50 | 64 | **0.466227** |
| 4 | 50 | 64 | **0.517638** |

Предсказанная последовательность символов Чесотки – 200, исходная строка – «Профилактика чесотки у человек» (30 символов), результат работы наилучшей модели продемонстрирован на рисунке 10.

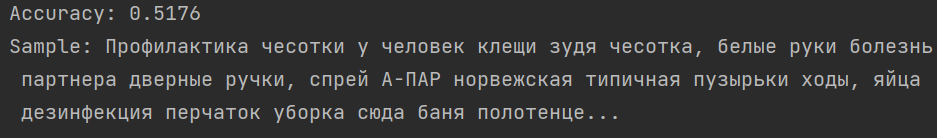


Рисунок 10 – Результат наилучшей GRU модели

Таблица 21 – Гиперпараметры LSTM сеть Шизофрения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | LSTM слой | Слой эмбендинга слов | Функция активации RNN | Функция активации Output |
| 1 | 128 | 128 | tanh | softmax |
| 2 | 128 | 128 | relu | softmax |
| 3 | 256 | 256 | tanh | softmax |
| 4 | 256 | 256 | relu | softmax |

Таблица 22 – Параметры обучения LSTM сеть Шизофрения

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Количество эпох | Размер бэтча | Accuracy |
| 1 | 25 | 64 | 0.322817 |
| 2 | 25 | 64 | 0.338594 |
| 3 | 50 | 64 | 0.491205 |
| 4 | 50 | 64 | 0.538376 |

Предсказанная последовательность символов Шизофрении – 200, исходная строка – «Шизофрения – одно из наиболее » (30 символов), результат работы наилучшей модели продемонстрирован на рисунке 11.

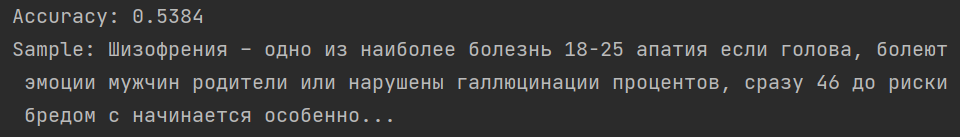


Рисунок 11 – Результат наилучшей LSTM модели

Таблица 23 – Гиперпараметры GRU сеть Шизофрения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | LSTM слой | Слой эмбендинга слов | Функция активации RNN | Функция активации Output |
| 1 | 128 | 128 | tanh | softmax |
| 2 | 128 | 128 | relu | softmax |
| 3 | 256 | 256 | tanh | softmax |
| 4 | 256 | 256 | relu | softmax |

Таблица 24 – Параметры обучения GRU сеть Шизофрения

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Количество эпох | Размер бэтча | Accuracy |
| 1 | 25 | 64 | 0.335672 |
| 2 | 25 | 64 | 0.363451 |
| 3 | 50 | 64 | 0.485893 |
| 4 | 50 | 64 | 0.519024 |

Предсказанная последовательность символов Шизофрении – 200, исходная строка – «Шизофрения – одно из наиболее » (30 символов), результат работы наилучшей модели продемонстрирован на рисунке 12.

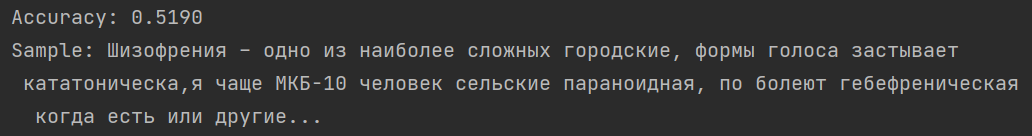


Рисунок 12 – Результат наилучшей GRU модели

Промежуточный вывод: В процессе обучения LSTM модель достигает в среднем лучших показателей метрики точности чем GRU модель. Показатель точности > 0.49 для большинства экспериментов является пограничным значением, когда модель становится способной к генерации текста, при меньших значениях данного показателя, в генерации текста наблюдалась зацикленность (Рисунок 13). Для генерации более осмысленного текста модели следует обучать больше.

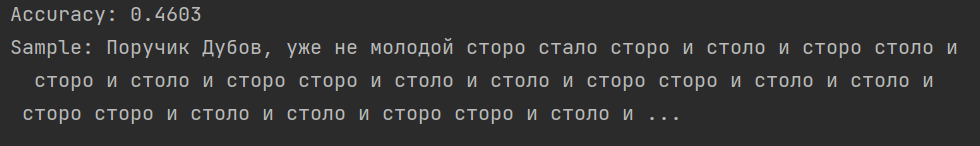


Рисунок 13 – Пример зацикленности при генерации текста

Задача 2: Прогнозирование наличия у пациента сахарного диабета с использованием LSTM и GRU моделей.

Таблица 25 – Разметка данных исходного датасета

|  |  |
| --- | --- |
| Параметр | Описание |
| Диабет\_012 | 0 = отсутствие диабета 1 = преддиабет 2 = сахарный диабет, |
| Высокий уровень (давление) | 0 = нет 1 = да |
| Высокий холестерин | 0 = нет 1 = да |
| Проверка холестерина | 0 = не проверял уровень холестерина в течение 5 лет 1 = проверял уровень холестерина в течение 5 лет |
| ИМТ | Индекс массы тела |
| Курильщик | 0 = нет 1 = да |
| Инсульт | 0 = нет 1 = да |
| Сердечное заболевание | 0 = нет 1 = да |
| Физическая активность | 0 = нет 1 = да |
| Фрукты | 0 = нет 1 = да |

Ключевой код инициализации LSTM модели представлен ниже:

|  |
| --- |
| def create\_lstm\_model():  model = Sequential([  LSTM(128, input\_shape=(X\_train.shape[1], X\_train.shape[2]), return\_sequences=True),  Dropout(0.2),  LSTM(64),  Dropout(0.2),  Dense(3, activation='softmax')  ])  model.compile(optimizer=Adam(learning\_rate=0.001),  loss='categorical\_crossentropy',  metrics=['accuracy'])  return model |

Функция потерь – CategoricalCrossentropy. Функция оптимизации – adam. Метрика точности – accuracy. Остальные гиперпараметры экспериментов и параметры обучения продемонстрированы в таблицах 1 и 2 соответственно.

Таблица 26 – Гиперпараметры LSTM сеть

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | LSTM слой 1 | LSTM слой 2 | Функция активации RNN | Функция активации Output |
| 1 | 64 | 32 | relu | softmax |
| 2 | 128 | 64 | relu | softmax |
| 3 | 64 | 32 | relu | softmax |
| 4 | 128 | 64 | relu | softmax |

Таблица 27 – Параметры обучения LSTM сеть

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Количество эпох | Размер бэтча | Accuracy |
| 1 | 25 | 32 | 0.8495 |
| 2 | 25 | 32 | 0.8481 |
| 3 | 50 | 32 | 0.8485 |
| 4 | 50 | 32 | 0.8459 |

Ключевой код инициализации GRU модели представлен ниже:

|  |
| --- |
| def create\_gru\_model():  model = Sequential([  GRU(128, input\_shape=(X\_train.shape[1], X\_train.shape[2]), return\_sequences=True),  Dropout(0.2),  GRU(64),  Dropout(0.2),  Dense(3, activation='softmax')  ])  model.compile(optimizer=Adam(learning\_rate=0.001),  loss='categorical\_crossentropy',  metrics=['accuracy'])  return model |

Функция потерь – CategoricalCrossentropy. Функция оптимизации – adam. Метрика точности – accuracy. Остальные гиперпараметры экспериментов и параметры обучения продемонстрированы в таблицах 1 и 2 соответственно.

Таблица 28 – Гиперпараметры GRU сеть

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | LSTM слой | Слой эмбендинга слов | Функция активации RNN | Функция активации Output |
| 1 | 64 | 32 | relu | softmax |
| 2 | 128 | 64 | relu | softmax |
| 3 | 64 | 32 | relu | softmax |
| 4 | 128 | 64 | relu | softmax |

Таблица 29 – Параметры обучения GRU сеть

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Количество эпох | Размер бэтча | Accuracy |
| 1 | 25 | 32 | 0.8490 |
| 2 | 25 | 32 | 0.8487 |
| 3 | 50 | 32 | 0.8501 |
| 4 | 50 | 32 | 0.8489 |

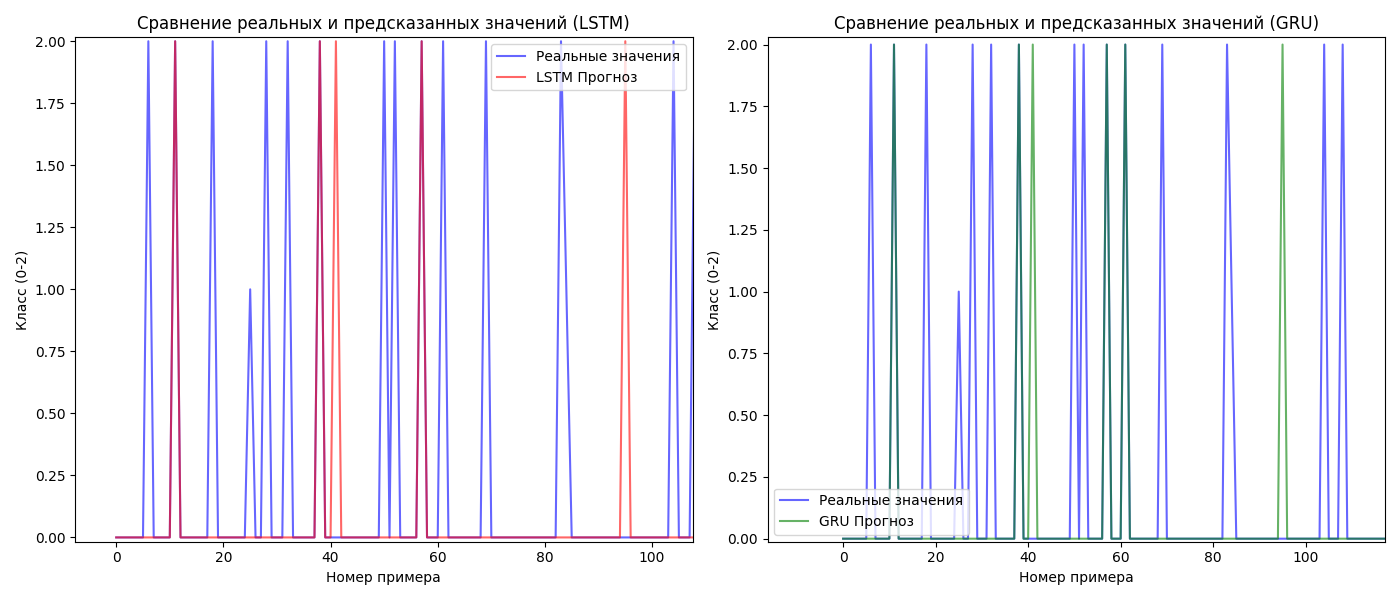


Рисунок 14 – Сравнение реальных и предсказанных значений для лучших LSTM и GRU моделей

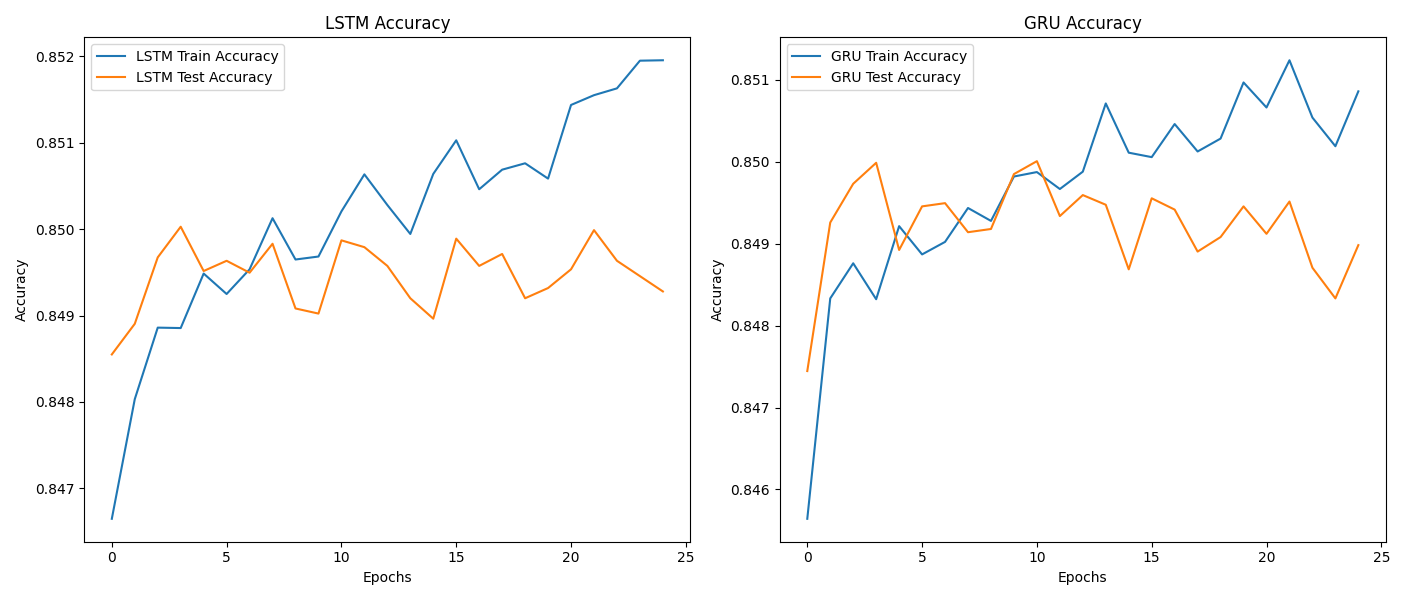


Рисунок 15 – Значения метрик точности моделей для экспериментов №1

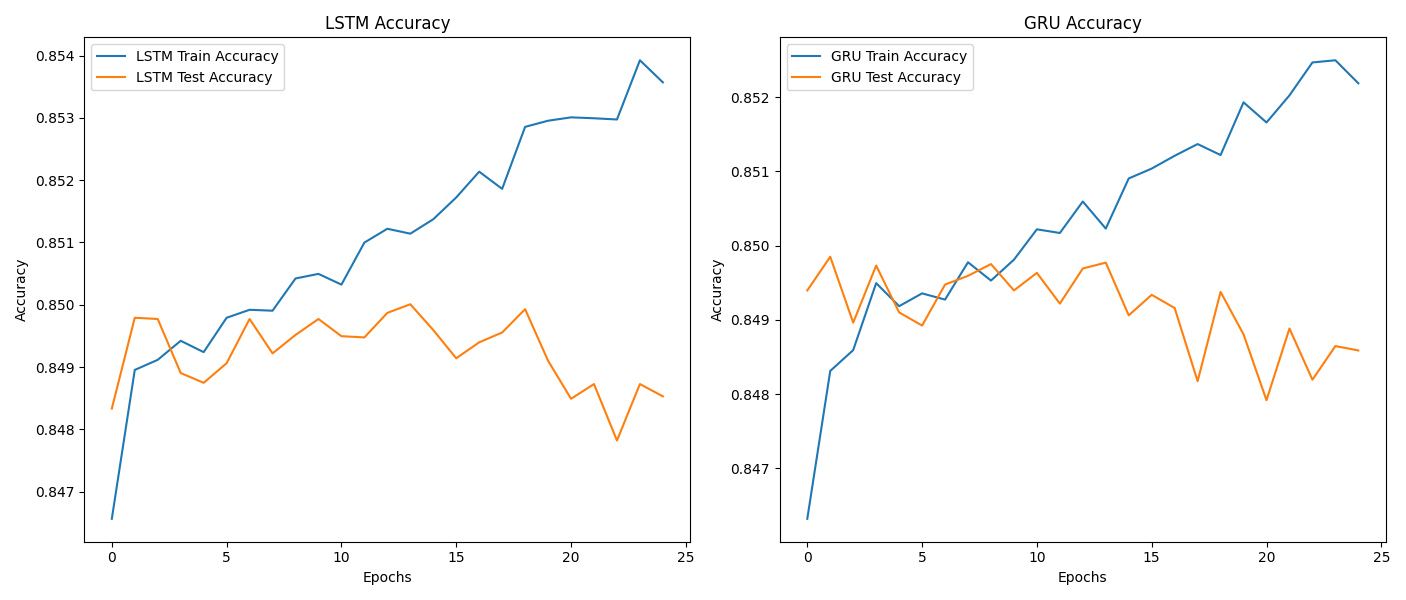


Рисунок 16 – Значения метрик точности моделей для экспериментов №2

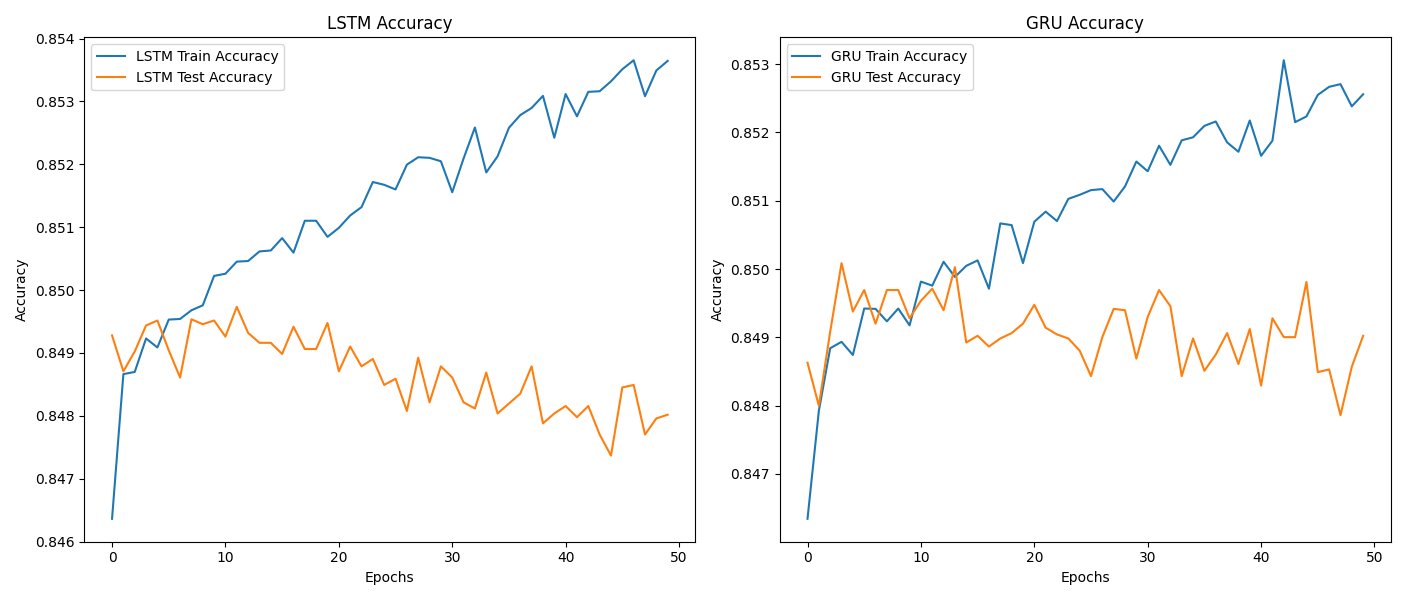


Рисунок 17 – Значения метрик точности моделей для экспериментов №3

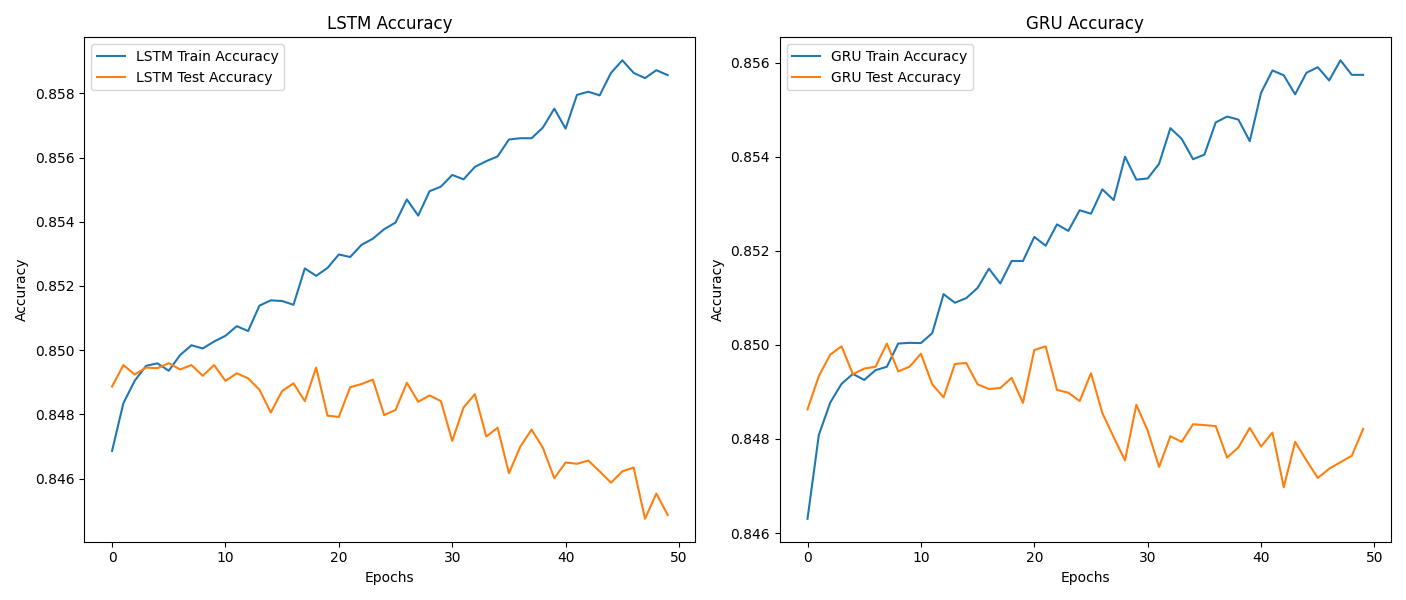


Рисунок 18 – Значения метрик точности моделей для экспериментов №4

Вывод: Входе выполнения практической работы были изучены принципы работы искусственных рекуррентных нейронных сетей, в частности моделей LSTM и GRU. Получены навыки обучения данных моделей для задач генерации текста и прогнозирования. Рассмотрены различные параметры настройки моделей и их влияние на точность и качество результатов. Таким образом поставленная цель достигнута, задание выполнено.

**Приложение A**

(обязательное)

Листинг файла task1v2.py

import numpy as np

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, GRU, Dense, Embedding

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.losses import SparseCategoricalCrossentropy

from tensorflow.keras.callbacks import ModelCheckpoint

import os

import pandas as pd

from datetime import datetime

# Загрузка и подготовка данных

def load\_text(filename):

with open(filename, 'r', encoding='utf-8') as file:

text = file.read()

return text

def prepare\_data(text, seq\_length):

chars = sorted(list(set(text)))

char\_to\_int = {ch: i for i, ch in enumerate(chars)}

int\_to\_char = {i: ch for i, ch in enumerate(chars)}

num\_chars = len(text)

num\_vocab = len(chars)

print(f"Total Characters: {num\_chars}")

print(f"Total Vocab: {num\_vocab}")

dataX = []

dataY = []

for i in range(0, num\_chars - seq\_length, 1):

seq\_in = text[i:i + seq\_length]

seq\_out = text[i + seq\_length]

dataX.append([char\_to\_int[char] for char in seq\_in])

dataY.append(char\_to\_int[seq\_out])

X = np.reshape(dataX, (len(dataX), seq\_length))

y = np.array(dataY)

return X, y, num\_vocab, char\_to\_int, int\_to\_char

# Создние модели LSTM с настраиваемой активацией

def create\_lstm\_model(seq\_length, num\_vocab, embedding\_dim, lstm\_units, lstm\_activation, output\_activation):

model = Sequential([

Embedding(num\_vocab, embedding\_dim, input\_length=seq\_length),

LSTM(lstm\_units, return\_sequences=False, activation=lstm\_activation, dropout=0.2, recurrent\_dropout=0.2),

# LSTM(lstm\_units,

# activation=lstm\_activation,

# input\_shape=(seq\_length, 1)),

# LSTM(lstm\_units, return\_sequences=False, activation=lstm\_activation),

Dense(num\_vocab, activation=output\_activation)

])

model.compile(

loss=SparseCategoricalCrossentropy(),

optimizer=Adam(learning\_rate=0.0001),

metrics=['accuracy']

)

return model

# Создание модели GRU с настраиваемой активацией

def create\_gru\_model(seq\_length, num\_vocab, embedding\_dim, gru\_units, gru\_activation, output\_activation):

model = Sequential([

Embedding(num\_vocab, embedding\_dim, input\_length=seq\_length),

GRU(gru\_units, return\_sequences=False, activation=gru\_activation, dropout=0.2, recurrent\_dropout=0.2),

# GRU(gru\_units,

# activation=gru\_activation,

# input\_shape=(seq\_length, 1)),

# GRU(gru\_units, return\_sequences=False, activation=gru\_activation),

Dense(num\_vocab, activation=output\_activation)

])

model.compile(

loss=SparseCategoricalCrossentropy(),

optimizer=Adam(learning\_rate=0.0001),

metrics=['accuracy']

)

return model

# Обучение модели

def train\_model(model, X, y, epochs, batch\_size, model\_name):

checkpoint\_path = f"Checkpoints/{model\_name}\_checkpoint.ckpt"

checkpoint\_dir = os.path.dirname(checkpoint\_path)

os.makedirs(checkpoint\_dir, exist\_ok=True)

cp\_callback = ModelCheckpoint(

filepath=checkpoint\_path,

save\_weights\_only=True,

)

history = model.fit(

X, y,

epochs=epochs,

batch\_size=batch\_size,

callbacks=[cp\_callback],

)

return history

# Генерация текста

def generate\_text(model, seed\_text, char\_to\_int, int\_to\_char, seq\_length, num\_chars=200):

generated = seed\_text

for \_ in range(num\_chars):

x = np.array([[char\_to\_int[char] for char in seed\_text[-seq\_length:]]])

pred = model.predict(x, verbose=0)

next\_char = int\_to\_char[np.argmax(pred)]

generated += next\_char

seed\_text += next\_char

return generated

# Эксперимент с разными гиперпараметрами

def run\_experiment(author, text, seq\_length, hyperparams, model\_type):

X, y, num\_vocab, char\_to\_int, int\_to\_char = prepare\_data(text, seq\_length)

results = []

for hp in hyperparams:

embedding\_dim = hp['embedding\_dim']

units = hp['units']

epochs = hp['epochs']

batch\_size = hp['batch\_size']

rnn\_activation = hp['rnn\_activation']

output\_activation = hp['output\_activation']

if model\_type == 'lstm':

model = create\_lstm\_model(

seq\_length, num\_vocab,

embedding\_dim, units,

rnn\_activation, output\_activation

)

elif model\_type == 'gru':

model = create\_gru\_model(

seq\_length, num\_vocab,

embedding\_dim, units,

rnn\_activation, output\_activation

)

model\_name = (f"{author}\_{model\_type}\_emb{embedding\_dim}\_units{units}\_"

f"rnnact{rnn\_activation}\_outact{output\_activation}\_"

f"ep{epochs}\_bs{batch\_size}")

print(f"\nTraining {model\_name}...")

history = train\_model(model, X, y, epochs, batch\_size, model\_name)

final\_accuracy = history.history['accuracy'][-1]

seed\_text = text[:seq\_length]

generated\_text = generate\_text(model, seed\_text, char\_to\_int, int\_to\_char, seq\_length)

result = {

'author': author,

'model\_type': model\_type,

'embedding\_dim': embedding\_dim,

'units': units,

'rnn\_activation': rnn\_activation,

'output\_activation': output\_activation,

'epochs': epochs,

'batch\_size': batch\_size,

'final\_accuracy': final\_accuracy,

'generated\_text\_sample': generated\_text[:200] + "...",

'timestamp': datetime.now().strftime("%Y-%m-%d %H:%M:%S")

}

results.append(result)

print(f"Accuracy: {final\_accuracy:.4f}")

print(f"Sample: {generated\_text[:200]}...")

return results

# Основная функция

def main():

authors = {

'chekhov': 'Чехов.txt',

# 'mayakovsky': 'Маяковский.txt',

# 'dostoevsky': 'Достоевский.txt'

'kor': 'Корь.txt',

# 'chesotka': 'Чесотка.txt',

# 'shizofrenia': 'Шизофрения.txt'

}

seq\_length = 30

# Доступные функции активации

rnn\_activations = [

'tanh',

'relu',

# 'sigmoid'

]

output\_activations = [

'softmax',

# 'linear',

]

# Гиперпараметры для перебора

base\_hyperparams = [

{'embedding\_dim': 128, 'units': 128, 'epochs': 25, 'batch\_size': 64},

{'embedding\_dim': 256, 'units': 256, 'epochs': 50, 'batch\_size': 64}

]

# Создаем все комбинации гиперпараметров с разными функциями активации

hyperparams = []

for base in base\_hyperparams:

for rnn\_act in rnn\_activations:

for out\_act in output\_activations:

hp = base.copy()

hp['rnn\_activation'] = rnn\_act

hp['output\_activation'] = out\_act

hyperparams.append(hp)

all\_results = []

for author, filename in authors.items():

print(f"\n--- Processing {author} ---")

text = load\_text(filename)

# Тестируем LSTM

lstm\_results = run\_experiment(author, text, seq\_length, hyperparams, 'lstm')

all\_results.extend(lstm\_results)

# Тестируем GRU

gru\_results = run\_experiment(author, text, seq\_length, hyperparams, 'gru')

all\_results.extend(gru\_results)

# Сохраняем результаты в CSV

df = pd.DataFrame(all\_results)

os.makedirs('results', exist\_ok=True)

results\_file = f"results/text\_generation\_results\_{datetime.now().strftime('%Y%m%d\_%H%M%S')}.csv"

df.to\_csv(results\_file, index=False)

print(f"\nResults saved to '{results\_file}'")

# Выводим сводную таблицу

print("\nSummary Table:")

print(df[['author', 'model\_type', 'embedding\_dim', 'units',

'rnn\_activation', 'output\_activation',

'epochs', 'batch\_size', 'final\_accuracy']].to\_string())

if \_\_name\_\_ == "\_\_main\_\_":

main()

Листинг файла task2.py

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler, LabelEncoder

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, GRU, Dense, Dropout

from tensorflow.keras.utils import to\_categorical

from tensorflow.keras.optimizers import Adam

# Загрузка данных (замените на ваш путь к файлу)

data = pd.read\_csv('diabetes\_012.csv') # Убедитесь, что файл имеет правильные поля

# Разделение на признаки и целевую переменную

X = data.drop('Diabetes\_012', axis=1).values

y = data['Diabetes\_012'].values

# Преобразование целевой переменной в one-hot encoding

y = to\_categorical(y, num\_classes=3)

# Разделение на обучающую и тестовую выборки (80/20)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42, stratify=y)

# Нормализация данных

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Преобразование данных для LSTM/GRU (добавление временного измерения)

X\_train = X\_train.reshape(X\_train.shape[0], 1, X\_train.shape[1])

X\_test = X\_test.reshape(X\_test.shape[0], 1, X\_test.shape[1])

# Создание модели LSTM

def create\_lstm\_model():

model = Sequential([

LSTM(64, input\_shape=(X\_train.shape[1], X\_train.shape[2]), return\_sequences=True),

Dropout(0.2),

LSTM(32),

Dropout(0.2),

Dense(3, activation='softmax')

])

model.compile(optimizer=Adam(learning\_rate=0.001),

loss='categorical\_crossentropy',

metrics=['accuracy'])

return model

# Создание модели GRU

def create\_gru\_model():

model = Sequential([

GRU(64, input\_shape=(X\_train.shape[1], X\_train.shape[2]), return\_sequences=True),

Dropout(0.2),

GRU(32),

Dropout(0.2),

Dense(3, activation='softmax')

])

model.compile(optimizer=Adam(learning\_rate=0.001),

loss='categorical\_crossentropy',

metrics=['accuracy'])

return model

# Обучение моделей

lstm\_model = create\_lstm\_model()

gru\_model = create\_gru\_model()

history\_lstm = lstm\_model.fit(X\_train, y\_train,

epochs=10,

batch\_size=32,

validation\_data=(X\_test, y\_test),

verbose=1)

history\_gru = gru\_model.fit(X\_train, y\_train,

epochs=10,

batch\_size=32,

validation\_data=(X\_test, y\_test),

verbose=1)

# Визуализация результатов

plt.figure(figsize=(14, 6))

# График точности LSTM

plt.subplot(1, 2, 1)

plt.plot(history\_lstm.history['accuracy'], label='LSTM Train Accuracy')

plt.plot(history\_lstm.history['val\_accuracy'], label='LSTM Test Accuracy')

plt.title('LSTM Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

# График точности GRU

plt.subplot(1, 2, 2)

plt.plot(history\_gru.history['accuracy'], label='GRU Train Accuracy')

plt.plot(history\_gru.history['val\_accuracy'], label='GRU Test Accuracy')

plt.title('GRU Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.tight\_layout()

plt.show()

# Получение предсказаний

y\_pred\_lstm = lstm\_model.predict(X\_test)

y\_pred\_gru = gru\_model.predict(X\_test)

# Преобразование one-hot обратно в классы

y\_test\_classes = np.argmax(y\_test, axis=1)

y\_pred\_lstm\_classes = np.argmax(y\_pred\_lstm, axis=1)

y\_pred\_gru\_classes = np.argmax(y\_pred\_gru, axis=1)

# 1. График реальных vs предсказанных значений (LSTM)

plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)

plt.plot(range(len(y\_test\_classes)), y\_test\_classes, color='blue', label='Реальные значения', alpha=0.6)

plt.plot(range(len(y\_pred\_lstm\_classes)), y\_pred\_lstm\_classes, color='red', label='LSTM Прогноз', alpha=0.6)

plt.title('Сравнение реальных и предсказанных значений (LSTM)')

plt.xlabel('Номер примера')

plt.ylabel('Класс (0-2)')

plt.legend()

# 2. График реальных vs предсказанных значений (GRU)

plt.subplot(1, 2, 2)

plt.plot(range(len(y\_test\_classes)), y\_test\_classes, color='blue', label='Реальные значения', alpha=0.6)

plt.plot(range(len(y\_pred\_gru\_classes)), y\_pred\_gru\_classes, color='green', label='GRU Прогноз', alpha=0.6)

plt.title('Сравнение реальных и предсказанных значений (GRU)')

plt.xlabel('Номер примера')

plt.ylabel('Класс (0-2)')

plt.legend()

plt.tight\_layout()

plt.show()

# Оценка моделей на тестовых данных

lstm\_score = lstm\_model.evaluate(X\_test, y\_test, verbose=0)

gru\_score = gru\_model.evaluate(X\_test, y\_test, verbose=0)

print(f"LSTM Test Accuracy: {lstm\_score[1]\*100:.2f}%")

print(f"GRU Test Accuracy: {gru\_score[1]\*100:.2f}%")