

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра математического обеспечения и применения ЭВМ**

**ОТЧЕТ**  
**по индивидуальному домашнему заданию**  
**«Перевод коротких фраз нейронной сетью»**  
**по дисциплине «Искусственные нейронные сети»**

Студент гр. 8382

Преподаватель

\_\_\_\_\_

\_\_\_\_\_

Облизов А.Д.

Жангиров Т.Р.

Санкт-Петербург

2021

## **Цель.**

Изучение применения искусственных нейронных сетей в области анализа текста и перевода предложений с одного языка на другой.

## **Задание.**

Требования к модели:

- Модель должна быть разработана на языке Python с использованием Keras API
- Исходный код проекта должен быть в формате PEP8
- В исходном коде должны быть поясняющие комментарии
- Модель не должна быть избыточной (должен соблюдаться баланс между размером сети [кол-во слоев и нейронов] и качеством выдаваемого результата)
- Обучение модели должно быть стабильно (для предложенной архитектуры ИНС обучение должно приводить к примерно одним и тем же результатам, то есть не должно быть такого, что при обучении 10 сетей удовлетворительный результат дают только 5 из них)
- *Плюсом будет анализ с использованием callback'a TensorBoard*
- *Плюсом будет разработка собственных callback'ов*
- *Плюсом будет создание модели из ансамбля ИНС*

Для изучения был выбран составлен датасет на основе сайта Tatoeba (<https://tatoeba.org/ru/downloads>) – открытый источник переводов фраз с различных языков. Необходимо реализовать модель, осуществляющую перевод английских коротких фраз (не более 8 слов) на немецкий.

## **Выполнение работы.**

Работа выполнялась на базе операционной системы Windows 10 в среде разработки PyCharm и в онлайн сервисе Google Colab.

## Описание датасета и решаемой задачи

С ресурса Taboeba был скачан файл с фразами на английском и немецком языках с помощью инструмента на сайте, представленного на рис. 1.

## Загрузки

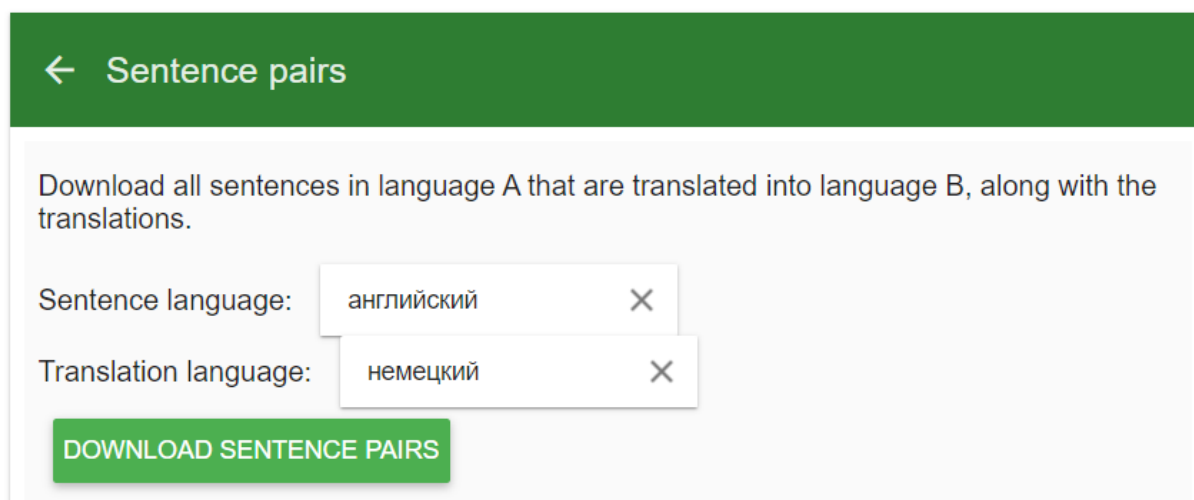


Рисунок 1 – Интерфейс загрузки наборов фраз

В файле содержалось более 150000 фраз разной длины и сложности. Было принято решение скорректировать датасет: были убраны фразы, содержащие более 8 слов, а также фразы с некоторыми именами. В результате был получен датасет, состоящий из 80000 фраз. Пример фрагмента датасета представлен на рис. 2.

36223	Nothing bad happened.	Nichts Schlimmes ist passiert.
36224	Nothing can stop him.	Nichts kann ihn aufhalten.
36225	Now listen carefully.	Jetzt hör gut zu.
36226	Now listen carefully.	Jetzt hören Sie gut zu.
36227	Now listen carefully.	Jetzt hört gut zu.
36228	Now we're in trouble.	Jetzt stecken wir in Schwierigkeiten.
36229	OK, I think I get it.	Okay, ich denke, ich verstehe das.
36230	Oil is running short.	Das Öl ist aufgebraucht.
36231	One of them is lying.	Einer von ihnen lügt.
36232	Oops, I did it again.	O nein! Ich hab's schon wieder getan!
36233	Open your mouth wide.	Den Mund bitte weit öffnen.
36234	Open your mouth wide.	Mach deinen Mund weit auf.
36235	Our dog seldom bites.	Unser Hund beißt nur selten.
36236	Our marriage is over.	Unsere Ehe endete.
36237	Our store isn't open.	Unser Geschäft ist nicht geöffnet.
36238	Our train is delayed.	Unser Zug hat Verspätung.
36239	People don't do that.	Man macht das nicht.
36240	People don't do that.	Die Leute machen das nicht.
36241	People here love you.	Die Leute hier lieben dich.
36242	People listen to Tom.	Die Leute hören auf Tom.
36243	People make mistakes.	Menschen machen Fehler.
36244	Perhaps he will come.	Vielleicht wird er kommen.

Рисунок 2 – Фрагмент датасета

Из фрагмента можно увидеть, что одни и те же фразы переводятся по-разному: это неизбежно, потому что перевод не является однозначным отображением.

Задача нейронной сети – обеспечить в большинстве случаев перевод пользовательских простых фраз на уровне, близком к уровню перевода онлайн-сервисами, без учета знаков препинания и регистра.

### Обработка данных

Подготовка данных для модели нейронной сети состоит из нескольких этапов:

- Форматирование текста: перевод всех буквенных символов в нижний регистр и удаление знаков препинания

С помощью такой операции можно сократить число уникальных слов (последовательностей символов), а также исключить влияние знаков препинания, которые пишутся слитно с некоторыми словами. Данная операция необходима для составления «словаря» слов на английском и немецком языках.

Листинг:

```
vocab[:, 0] = [s.translate(str.maketrans('', '', string.punctuation)) for s in vocab[:, 0]]
vocab[:, 1] = [s.translate(str.maketrans('', '', string.punctuation)) for s in vocab[:, 1]]

for i in range(len(vocab)):
    vocab[i, 0] = vocab[i, 0].lower()
    vocab[i, 1] = vocab[i, 1].lower()
```

- Превращение фраз в токены, представляющих собой массивы индексов.

Сеть работает с числами, а не со словами, поэтому используется встроенный в Keras объект `Tokenizer`. С помощью данного объекта производится индексация каждого уникального слова во входных данных и в переводе. Далее каждая фраза трансформируется в массив из 8 индексов слов в том же порядке, в каком слова расположены в этой фразе. Если фраза содержит менее 8 слов, то в конце массив заполнен нулями. Листинг настройки объекта `Tokenizer` для входных данных приведен в листинге ниже.

Листинг:

```
data_tokenizer = Tokenizer()
data_tokenizer.fit_on_texts(vocab[:, 0])
data_vsize = len(data_tokenizer.word_index) + 1
data_tsize = 8
...
def encode_sequences(tokenizer, length, lines):
    seq = tokenizer.texts_to_sequences(lines)
    seq = pad_sequences(seq, maxlen=length, padding='post')
    return seq
...
trainX = encode_sequences(data_tokenizer, data_tsize, train[:, 0])
trainY = encode_sequences(label_tokenizer, label_tsize, train[:, 1])
```

Пример изначальной фразы и ее токена:

- Фраза: how long is it
- Токен: [ 47 182 4 8 0 0 0 0]

## Создание модели

Так как перевод не является отображением каждого слова из один язык в другой, а подразумевает учет контекста, было решено использовать рекуррентную модель нейронной сети.

Итоговая модель нейронной сети представлена в табл. 1.

Таблица 1 – Структура модели

Слой	Особенности	Выход	Число парам.
Embedding	Индекс 0 не используется в словаре (mask_zero=True)	(None, 8, 512)	3618816
LSTM	Число ячеек – 512	(None, 512)	2099200
Dropout	Коэффициент – 0.3	(None, 512)	0
RepeatVector	Число повторений – 8	(None, 8, 512)	0
LSTM	Возвращает всю последовательность (return_sequences=True)	(None, 8, 512)	2099200
Dropout	Коэффициент – 0.3	(None, 8, 512)	0
Dense	8 нейр., функция активации – softmax	(None, 8, 12159)	6237567

Изменения, которые были внесены в модель при исследовании:

- Были добавлены слои Dropout, так как модель была склонна к быстрому переобучению.
- Число ячеек слоев LSTM варьировалось от 300 до 700, в результате было выбрано число 512, как обеспечивающее лучшее сочетание скорости обучения (вычислительного), точности результата и отсутствию переобучения.

Параметры компиляции модели:

- Был выбран оптимизатор RMSProp с установленной скоростью обучения 0.001. В данной задаче оптимизатор Adam показывал более низкую скорость обучения и приводил к переобучению при большем значении потерь.
- Функция потерь: sparse\_categorical\_crossentropy. Для задачи обработки текста с большим количеством уникальных слов подходит

лучше, чем функция категориальной кросс-энтропии, так как учитывает именно целевой индекс слова, который должна рассчитать модель.

$$CCE(p, t) = - \sum_{c=1}^c t_{o,c} \log(p_{o,c})$$

Параметры обучения модели:

- Число эпох – 30
- Размер батча – 256

Были испробованы разные размеры батча: более низкие не позволяли точно контролировать точку переобучения, а более высокие давали более высокое значение ошибок модели

- Данные для валидации: 20% от тренировочного набора
- Callbacks
  - TensorBoard для сбора статистики обучения модели
  - EarlyStopping для остановки модели в случае переобучения (параметр для слежения – потери на данных для валидации, ожидание – 3 эпохи)
  - ModelCheckpoint для сохранения лучшей модели (параметр для выбора – потери на данных для валидации)
  - ReduceLROnPlateau для снижения скорости обучения, когда потери на данных для валидации перестают уменьшаться. Ожидание – 2 эпохи, коэффициент (factor) – 0.1.
  - Собственный Callback TranslateEveryFive, который осуществляет тестирование на пользовательских данных раз в 5 эпох.

```
class TranslateEveryFive(keras.callbacks.Callback):  
    def __init__(self):  
        super(MyCustomCallback, self).__init__()  
  
    def on_epoch_end(self, epoch, logs=None):
```

```

if epoch % 5 == 0:
    translate(self.model, custom_data, custom_text, epoch + 1)

```

## Результаты обучения и тестирования.

График потерь взят из веб-приложения TensorBoard и приведен на рис. 3.

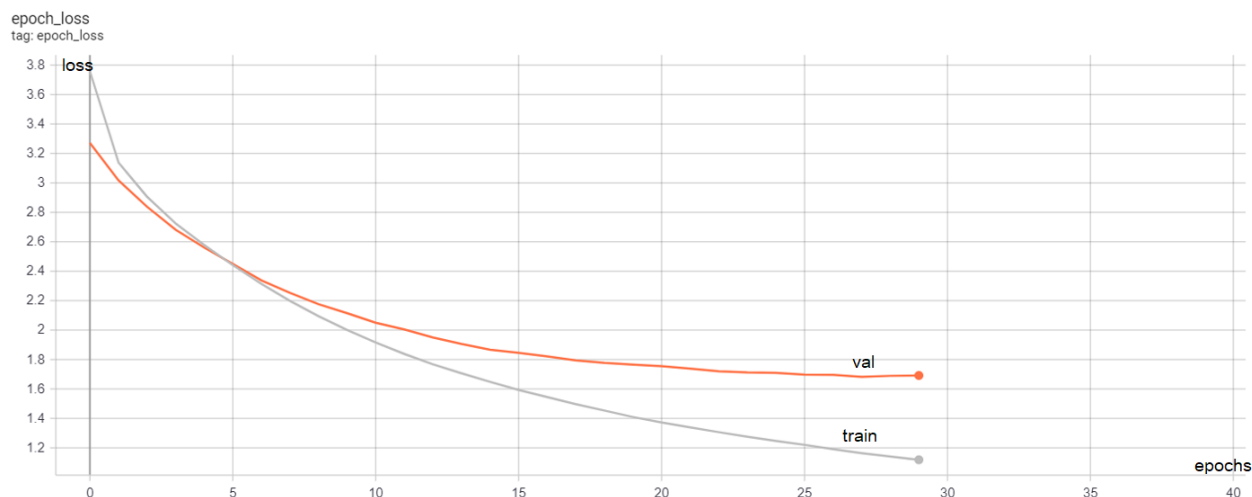


Рисунок 3 – График потерь

Стоит отметить, что точность модели определяется по корректному предсказанию всего предложения (то есть, что модель перевела фразу в точности так, как это указано в датасете). Данный показатель не является решающим, потому что одно и то же слово может иметь синонимы, и даже в датасете есть одинаковые фразы на английском языке, которые по-разному переводятся на немецкий.

График точности также из TensorBoard приведен на рис. 4.

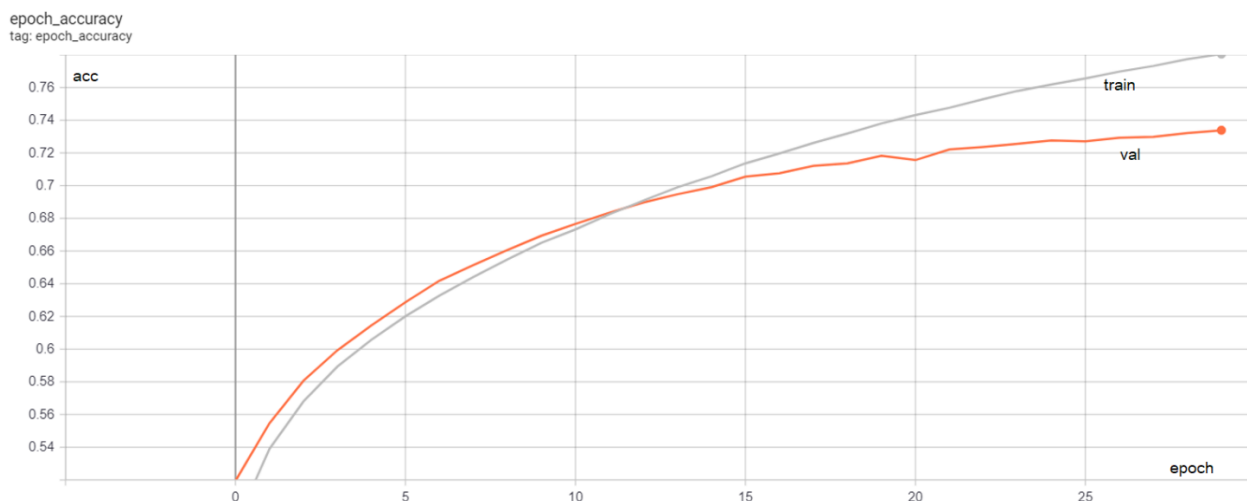


Рисунок 4 – График точности



После обучения модель на тестовых данных достигла следующих показателей:

- Точность: 74.2%
- Потери: 1.637

На рис. 5 приведен график ошибок относительно итераций (батчей).

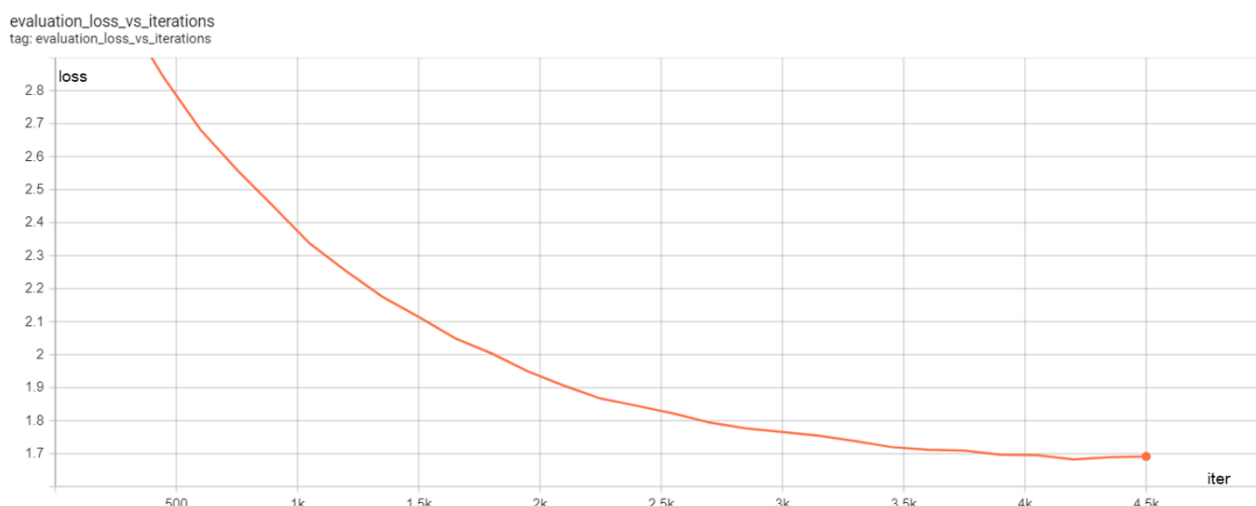


Рисунок 5 – График ошибок относительно итераций (батчей)

Гистограммы смещений (bias), ядра одного из слоев LSTM (kernel\_0) и рекуррентного ядра слоя LSTM (reccurent\_kernel\_0) представлены на рис. 6.

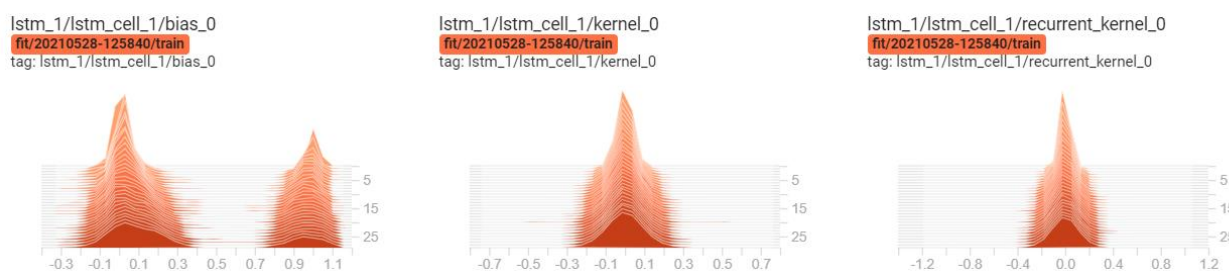


Рисунок 6 – Гистограммы для слоя LSTM

Из гистограмм видно, что в начале обучения графики имели выраженные пики, которые с ходом обучения разглаживались. У смещений есть два выраженных интервала:  $[-0.1, 0.3]$ ,  $[0.8, 1.1]$ . Распределение весов практически симметрично относительно 0 на последних эпохах. Для другого LSTM слоя графики схожи.

Гистограммы смещений (bias\_0) и весов (kernel\_0) слоя Dense представлены на рис. 7.

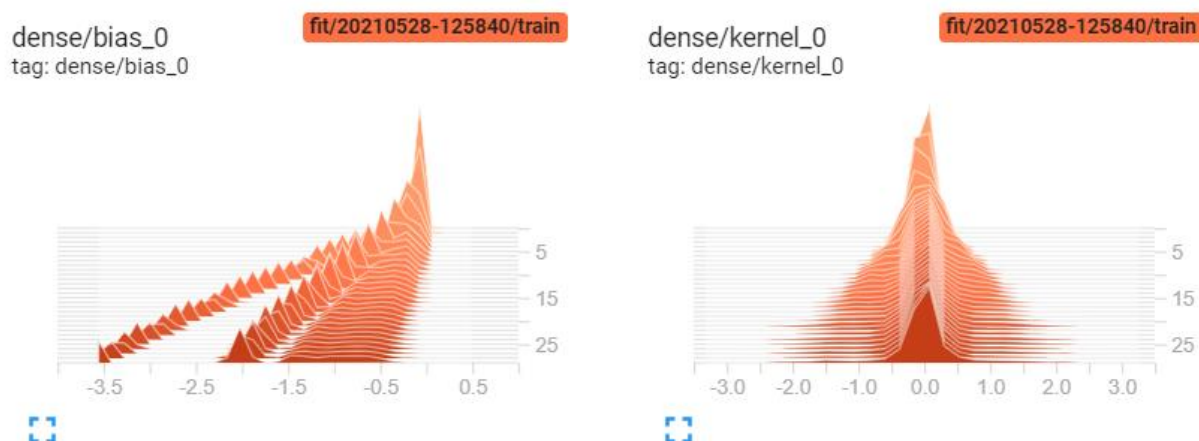


Рисунок 7 – Гистограммы слоя Dense

Из гистограмм видно, что слои обучаются уже к 10 эпохе, а дальнейшие изменения несущественны.

Для обратного преобразования выхода нейронной сети (токена размером 8) была создана функция `get_word`, листинг которой приведен ниже:

```
def get_word(n, tokenizer):
    if n == 0:
        return ""
    for word, index in tokenizer.word_index.items():
        if index == n:
            return word
    return ""
```

В табл. 2 приведены пользовательские фразы, их перевод с помощью переводчика Google и с помощью нейронной сети (после обучения).

Таблица 2 – Сравнение переводов

Оригинал	Google-переводчик	Предсказание	Комментарий
i want to eat an apple	Ich möchte einen Apfel	ich möchte einen apfel	Полное соответствие
my name is tom	mein Name ist Tom	mein name ist tom	Полное соответствие
how old are you	wie alt bist du	wie alt sind sie	Нейронная сеть применила местоимение «ВЫ», а переводчик – «ТЫ».
where is the bathroom	Wo ist die Toilette	wo ist das toilette	Полное соответствие

i really like you	Ich mag dich wirklich	ich mag dich wirklich	Полное соответствие
life is so hard	das Leben ist so hart	das leben ist so hart	Полное соответствие

Как видно из пользовательских фраз, нейронная сеть отлично переводит простые фразы на немецкий язык.

В табл. 3 приведены переводы нейронной сети во время обучения, полученные с помощью TranslateEveryFive.

Таблица 3 – Переводы во время обучения (Callback)

Эпоха	i want to eat an apple	my name is tom	how old are you	where is the bathroom	i really like you	life is so hard
1	ich habe nicht nicht	sie ist das	er ist es nicht	sie ist ist	ich habe nicht nicht	er ist es nicht
6	ich muss einen einen	mein vater ist tom	wie sind sind sie	wo ist dein	ich habe dich nicht	das ist ist nicht
11	ich möchte ein paar	mein hund ist tom	wie alt sind sie	wo ist die	ich habe dich dich	das leben ist nicht
16	ich möchte einen apfel	mein name ist tom	wie alt sind sie	wo ist das toilette	ich mag dich dich dich	das leben ist nicht schwer
21	ich möchte einen apfel	mein vater ist tom	wie alt sind sie	wo ist das toilette	ich mag dich wirklich	das leben ist so schwer
26	ich möchte einen apfel	mein name ist tom	wie alt sind sie	wo ist das toilette	ich mag dich wirklich	das leben ist so schwer

Из таблицы видно, что на первых эпохах нейронная сеть выучила мало слов, которые повторяются почти в каждой из фраз. Далее сеть обучилась выявлять большее число слов и к 26-й эпохе практически идеально переводить простые фразы. Итоговой эпохой является 30-я, но дальнейшие изменения не повлияли на выход нейронной сети.

При нескольких прогонах сеть показывает себя крайне стабильно, отличия в переводе наблюдаются в некоторых словах: иногда используются синонимы, иногда имеют место ошибки, но в целом модель всегда достигает точности около 73%.

Результаты тестирования модели на более сложных фразах приведены в табл. 4.

Таблица 4 – Сравнение переводов (сложные фразы)

Оригинал	Google-переводчик	Предсказание	Комментарий
i don't want to use this tool	Ich möchte dieses Tool nicht verwend	ich möchte das für machen	Не было переведено слово «tool»
do you know how to get there	Weißt du, wie man dort hinkommt	weißt du dass man hier	Фраза искажена, но смысл улавливается
this film is terrible	Dieser Film ist schrecklich	dieser film ist furchtbar	Предсказание верно, слова-синонимы
i will borrow money to pay	Ich werde Geld leihen um zu bezahlen	ich werde mir mein	Слова «money», «pay» не переведены

Из результатов видно, что нейронная сеть не смогла корректно перевести все фразы: некоторые слова оказались ей неизвестными, а также возникли сложности с построением более длинных фраз.

Основные проблемы, которые были решены:

- Быстрое переобучение модели было решено добавлением слоев Dropout и подбором размера батча
- Низкая точность не оказалась существенной проблемой, так как при изучении датасета было выявлено, что используются слова-синонимы, и в целом перевод можно считать неточной задачей.

Проблемы, которые не были в полной мере решены:

- Небольшой словарный запас, который обусловлен простым датасетом. Для большего разнообразия слов требуется намного более объемный датасет, так как желательно, чтобы слово фигурировало в нескольких фразах.
- Не учитывается пунктуация: ее ввод сильно ухудшал показатели модели и было решено от нее избавляться.

Из этого можно определить следующие рекомендации для развития модели:

- Увеличение датасета для большего словарного запаса нейронной сети
- Использование структуры автоэнкодера для определения моделью более сложных конструкций
- Учет пунктуации для перевода более сложных фраз
- Применение обычных методов перевода (например, словарей слов) для проверки корректности перевода модели и корректировки.

### **Выводы.**

В ходе выполнения индивидуального задания было изучено применение нейронных сетей для перевода фраз с одного языка на другой. Была спроектирована и обучена модель, которая осуществляет перевод коротких простых фраз с английского языка на немецкий, проведен анализ результатов и тестирование на пользовательских фразах. В результате было сделан вывод, что рекуррентная модель способна корректно переводить простые фразы, но для перевода более сложных или длинных фраз требуется большое количество входных данных и усложненная структура модели.

## ПРИЛОЖЕНИЕ А

### Исходный код программы. Файл idz.py

```
import string
import datetime
import numpy as np
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense, LSTM, Embedding, RepeatVector, Dropout
from keras.preprocessing.text import Tokenizer
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau,
TensorBoard
from keras.preprocessing.sequence import pad_sequences
from keras.models import load_model
from keras import optimizers
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

def read_text(filename):
    with open(filename, mode='rt', encoding='utf-8') as file:
        text = file.read()
        phrases = text.strip().split('\n')
        return [phrase.split('\t') for phrase in phrases]

print("Loading file...")
data = read_text("vocab.txt")
vocab = np.array(data)

vocab = vocab[:60000, :]
print("Dictionary size:", vocab.shape)

print("Formatting text...")
vocab[:, 0] = [s.translate(str.maketrans('', '', string.punctuation)) for s in
vocab[:, 0]]
vocab[:, 1] = [s.translate(str.maketrans('', '', string.punctuation)) for s in
vocab[:, 1]]

for i in range(len(vocab)):
    vocab[i, 0] = vocab[i, 0].lower()
    vocab[i, 1] = vocab[i, 1].lower()

print("Tokenize...")
```

```

data_tokenizer = Tokenizer()
data_tokenizer.fit_on_texts(vocab[:, 0])
data_vsize = len(data_tokenizer.word_index) + 1
data_tsize = 8

label_tokenizer = Tokenizer()
label_tokenizer.fit_on_texts(vocab[:, 1])
label_vsize = len(label_tokenizer.word_index) + 1
label_tsize = 8

def encode_sequences(tokenizer, length, lines):
    seq = tokenizer.texts_to_sequences(lines)
    seq = pad_sequences(seq, maxlen=length, padding='post')
    return seq

def get_word(n, tokenizer):
    if n == 0:
        return ""
    for word, index in tokenizer.word_index.items():
        if index == n:
            return word
    return ""

def translate(model, custom_text, custom_answers, epoch = -1):
    custom_data = encode_sequences(data_tokenizer, data_tsize, custom_text)
    prediction = model.predict_classes(custom_data)
    if (epoch > 0):
        print("Epoch" + str(epoch))
    for j, pred in enumerate(prediction):
        print("Original:")
        print(custom_text[j])
        print("Google translate:")
        print(custom_answers[j])
        print("Prediction:")
        output = ""
        for i in range(len(pred)):
            if pred[i] != 0:
                output += str(get_word(pred[i], label_tokenizer)) + " "
            else:
                break
        print(output)
        print()

```

```

custom_text = ["i don't want to use this tool", "do you know how to get there",

```

```

        "this film is terrible", "i will borrow money to pay", "life is so
hard"]
custom_answers = ["Ich möchte dieses Tool nicht verwende", "Weißt du, wie man
dort hinkommt",
        "Dieser Film ist schrecklich", "Ich werde Geld leihen um zu
bezahlen", "das Leben ist so hart"]

train, test = train_test_split(vocab, test_size=0.2, random_state=12)

trainX = encode_sequences(data_tokenizer, data_tsize, train[:, 0])
trainY = encode_sequences(label_tokenizer, label_tsize, train[:, 1])

testX = encode_sequences(data_tokenizer, data_tsize, test[:, 0])
testY = encode_sequences(label_tokenizer, label_tsize, test[:, 1])

def make_model(in_vocab, out_vocab, in_timesteps, out_timesteps, n):
    model = Sequential()
    model.add(Embedding(in_vocab, n, input_length=in_timesteps, mask_zero=True))
    model.add(LSTM(n))
    model.add(Dropout(0.3))
    model.add(RepeatVector(out_timesteps))
    model.add(LSTM(n, return_sequences=True))
    model.add(Dropout(0.3))
    model.add(Dense(out_vocab, activation='softmax'))
    model.compile(optimizer=optimizers.RMSprop(learning_rate=0.001),
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    model.summary()
    return model

class TranslateEveryFive(keras.callbacks.Callback):
    def __init__(self):
        super(TranslateEveryFive, self).__init__()

    def on_epoch_end(self, epoch, logs=None):
        if epoch % 5 == 0:
            translate(self.model, custom_text, custom_answers, epoch + 1)

print("Data vocab size / token size:", data_vsize, data_tsize)
print("Label vocab size / token size:", label_vsize, label_tsize)

print("Input 0 if you want to load model or anything to fit model")
s = input()
if s is not "0":
    print("Initializing model...")
    model = make_model(data_vsize, label_vsize, data_tsize, label_tsize, 512)

```



```

log_dir="logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
num_epochs = 30
callbacks_list = [
    EarlyStopping(
        monitor='val_loss',
        patience=3,
    ),
    ModelCheckpoint(
        filepath='vocab-model.h5',
        monitor='val_loss',
        save_best_only=True,
    ),
    ReduceLROnPlateau(
        monitor='val_loss',
        factor=0.1,
        patience=2,
    ),
    TensorBoard(log_dir=log_dir, histogram_freq=1, embeddings_freq=1),
    TranslateEveryFive()
]

print("Training...")
history = model.fit(trainX, trainY.reshape(trainY.shape[0], trainY.shape[1],
1),
                    epochs=num_epochs, batch_size=256,
                    validation_split=0.2, callbacks=callbacks_list, verbose=1)
model.evaluate(testX, testY.reshape(testY.shape[0], testY.shape[1], 1))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['train', 'validation'])
plt.show()

model = load_model('vocab-model.h5')
translate(model, custom_text, custom_answers)

```