МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования «Московский технический университет связи и информатики»

Кафедра «Математическая кибернетика и информационные технологии»

Лабораторная работа № 7

тематика

Классификация обзоров фильмов

Москва 2021

Цель

Классификация последовательностей - это проблема прогнозирующего моделирования, когда у вас есть некоторая последовательность входных данных в пространстве или времени, и задача состоит в том, чтобы предсказать категорию для последовательности.
Проблема усложняется тем, что последовательности могут различаться по длине, состоять из очень большого словарного запаса входных символов и могут потребовать от модели изучения долгосрочного контекста или зависимостей между символами во входной последовательности.
В данной лабораторной работе также будет использоваться датасет IMDb, однако обучение будет проводиться с помощью рекуррентной нейронной сети.

Задачи

- Ознакомиться с рекуррентными нейронными сетями
- Изучить способы классификации текста
- Ознакомиться с ансамблированием сетей
- Построить ансамбль сетей, который позволит получать точность не менее 97%

Выполнение работы

Мы отобразим каждый обзор фильма в реальную векторную область, популярную технику при работе с текстом, которая называется встраивание слов. Это метод, в котором слова кодируются как действительные векторы в многомерном пространстве, где сходство между словами в смысле смысла приводит к близости в векторном пространстве.
Keras предоставляет удобный способ преобразования положительных целочисленных представлений слов в вложение слов слоем Embedding.
Мы сопоставим каждое слово с вещественным вектором длиной 32. Мы также ограничим общее количество слов, которые нам интересны в моделировании, до 5000 наиболее часто встречающихся слов и обнуляем остальные. Наконец, длина последовательности (количество слов) в каждом обзоре варьируется, поэтому мы ограничим каждый обзор 500 словами, укорачивая длинные обзоры и дополняя короткие обзоры нулевыми значениями.
Теперь, когда мы определили нашу проблему и то, как данные будут подготовлены и смоделированы, мы готовы разработать модель LSTM.

Импорт зависимостей и получение данных

Начнем с импорта необходимых зависимостей для предварительной обработки данных и построения модели.

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

Загрузим датесет IMDb, который уже встроен в Keras. Поскольку мы не хотим иметь данные обучения и тестирования в пропорции 50/50, мы сразу же объединим эти данные после загрузки для последующего разделения в пропорции 80/20:

```
from keras.datasets import imdb
(training_data, training_targets), (testing_data,
testing_targets) = imdb.load_data(num_words=10000)
data = np.concatenate((training_data, testing_data), axis=0)
```

```
targets = np.concatenate((training_targets, testing_targets),
axis=0)
```

Далее нам нужно обрезать и дополнить входные последовательности так, чтобы они были одинаковой длины для моделирования. Модель узнает, что нулевые значения не содержат никакой информации, поэтому на самом деле последовательности имеют разную длину с точки зрения содержания, но для выполнения вычислений в Керасе требуются векторы одинаковой длины.

```
max_review_length = 500
X_train = sequence.pad_sequences(X_train,
maxlen=max_review_length)
X_test = sequence.pad_sequences(X_test,
maxlen=max_review_length)
```

Архитектура сети

Первый слой - это Embedded, который использует 32 вектора длины для представления каждого слова. Следующий уровень - это слой LSTM с 100 единицами памяти (умными нейронами). Наконец, поскольку это проблема классификации, мы используем плотный выходной слой с одним нейроном и сигмоидной функцией активации, чтобы сделать 0 или 1 прогноз для двух классов (хорошего и плохого) в задаче.

Поскольку это проблема двоичной классификации, в качестве функции потерь используется журнал потерь (binary_crossentropy в Керасе). Используется эффективный алгоритм оптимизации ADAM. Модель подходит только для 2 эпох, потому что она быстро решает проблему. Большой пакет из 64 обзоров используется для разметки обновлений веса.

```
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length,
input_length=max_review_length))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
print(model.summary())
model.fit(X_train, y_train, validation_data=(X_test, y_test),
epochs=3, batch_size=64)
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Вы можете видеть, что этот простой LSTM с небольшими настройками достигает почти современных результатов по проблеме IMDB. Важно, что это шаблон, который вы можете использовать для применения сетей LSTM к вашим собственным задачам классификации последовательностей.

Рассмотрим некоторые расширения этой простой модели.

LSTM и сверточная нейронная сеть для классификации последовательностей

Сверточные нейронные сети превосходно изучают пространственную структуру во входных данных.

Данные обзора IMDB действительно имеют одномерную пространственную структуру в последовательности слов в обзорах, и CNN может быть в состоянии выбрать инвариантные особенности для хорошего и плохого настроения. Эти изученные пространственные особенности могут затем быть изучены как последовательности уровнем LSTM.

Мы можем легко добавить одномерный слой CNN и максимальный пул после слоя Embedding, которые затем передают объединенные элементы в LSTM. Мы можем

использовать небольшой набор из 32 объектов с небольшой длиной фильтра 3. Слой пула может использовать стандартную длину 2, чтобы вдвое уменьшить размер карты объектов. Например, мы бы создали модель следующим образом:

```
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length,
input_length=max_review_length))
model.add(Conv1D(filters=32, kernel_size=3, padding='same',
activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
```

Мы видим, что мы достигаем результатов, аналогичных первому примеру, но с меньшими весами и меньшим временем обучения.
Рекуррентные нейронные сети, такие как LSTM, обычно имеют проблему переобучения. Вам необходимо решить эту проблему путем добавления в архитектуру сети слоев Dropout.

Требования

1. Найти набор оптимальных ИНС для классификации текста
2. Провести ансамблирование моделей
3. Написать функцию/функции, которые позволят загружать текст и получать результат ансамбля сетей
4. Провести тестирование сетей на своих текстах (привести в отчете)