МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

ОТЧЕТ

по лабораторной работе №4

по дисциплине

Предиктивная аналитика

РУКОВОДИТЕЛЬ:

_____ Санников А.Н.

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

_____ Напылов Е.И.

(подпись) (фамилия, и.,о.)

_М22-ИВТ-1_____

(шифр группы)

Работа защищена «___» _____

С оценкой _____

Нижний Новгород 2023

Содержание

1. Постановка задачи

В данной работе необходимо решить задачу NLP, которая заключается в классификации текстов. Подобные задачи лучше всего решаются с помощью нейронных сетей с рекуррентной архитектурой.

Задача заключается в классификации постов пользователей “форума” 2ch.hk по разделам. Собранный парсером датасет содержит более 127000 сообщений из разделов программирование, столовая, фильмы, животные и природа, наука. В среднем сообщения содержат от 5 до 200 слов.

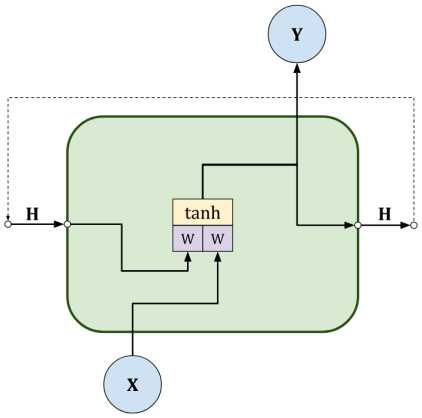
2. Рекуррентные нейронные сети

Рекуррентные нейронные сети (RNN) – это тип нейронных сетей, который может работать с последовательностями данных разной длины и хранить информацию о предыдущих состояниях внутри сети. Это позволяет использовать RNN для обработки временных рядов, текстов и других последовательностей данных.

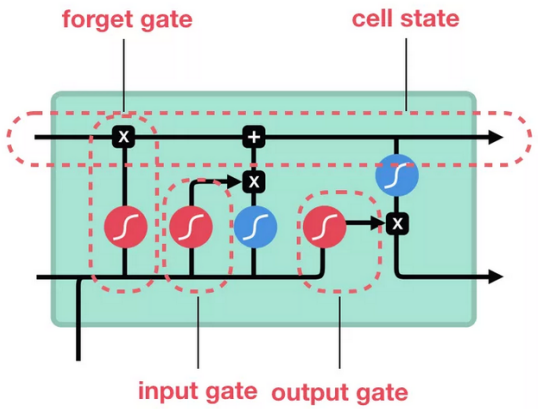
Одним из главных преимуществ RNN является возможность моделирования долгосрочных зависимостей в данных. Это достигается благодаря наличию в RNN циклов обратной связи, которые позволяют передавать информацию от предыдущего состояния к следующему. Каждое следующее состояние зависит от текущего входа и предыдущего состояния, что позволяет RNN учитывать контекст и последовательность данных.

Существуют несколько видов рекуррентных нейронных сетей.

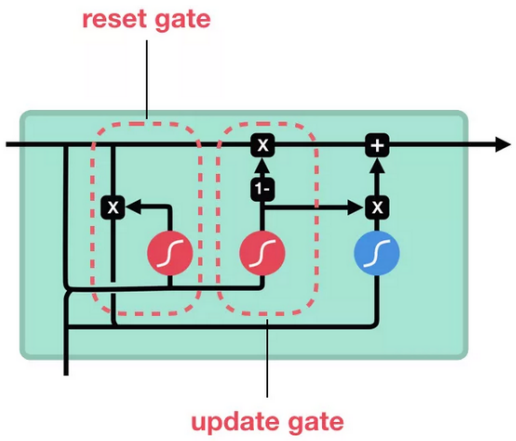
Простые рекуррентные нейронные сети (Simple RNN) – это наиболее простой тип RNN, который состоит из одного слоя рекуррентных нейронов и использует простую функцию активации, такую как гиперболический тангенс. Простые RNN могут работать с короткими последовательностями данных, но могут иметь проблемы с обработкой длинных последовательностей.



Нейронные сети долгой краткосрочной памяти (Long Short-Term Memory, LSTM) – это тип RNN, который использует специальные блоки памяти для хранения информации о предыдущих состояниях и предотвращения проблем с затухающими градиентами. Блоки LSTM могут добавлять или удалять информацию из памяти и фильтровать ее, что позволяет LSTM обрабатывать более длинные последовательности данных, чем простые RNN.



Сети GRU (Gated Recurrent Unit) – это еще один тип RNN, который использует механизмы блоков памяти, но в отличие от LSTM, использует меньше параметров и может быть быстрее в обучении. GRU имеет два блока памяти – блок обновления и блок сброса, которые определяют, какую информацию передавать дальше и какую игнорировать.



GRU, по сравнению с LSTM, чуть меньше подвержен переобучению, при этом имеет меньшую вычислительную сложность.

При стандартном поведении рекуррентные нейроны возвращают свою последнюю последовательность, однако, такой подход не является самым эффективным. Имеется возможность получить все последовательности одновременно и самостоятельно их обработать.

3. Данные и их обработка

С помощью самодельного парсера были собраны посты из 5-ти разделов форума. Посты были поделены на слова.

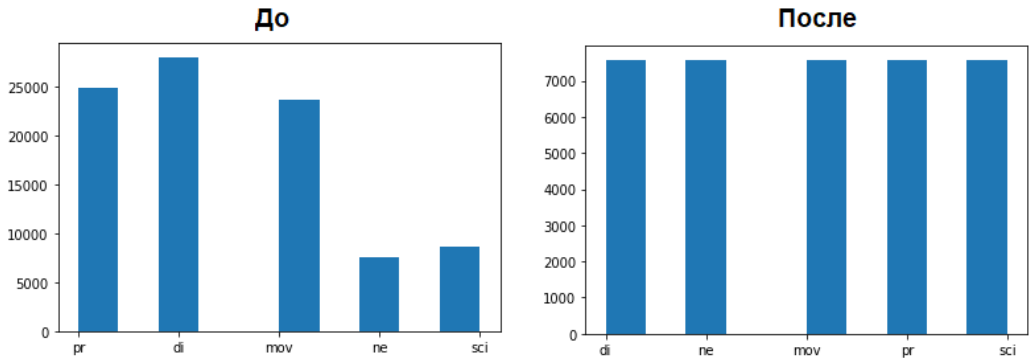
Обработка данных.

Данные в исходном виде совершенно не пригодны для обучения нейронной сети, поэтому их нужно почистить. Первый этап обработки:

1. Удаление тэгов местной разметки (гибрид html+markdown).
2. Удаление предлогов, числительных и т.п. Выполняется с помощью nltk.corpus.stopwords.
3. Удаление слишком коротких сообщений (< 5 слов).
4. Перевод всех слов в начальную форму. Выполняется с помощью pymorphy2.

После данного этапа посты выглядят приблизительно так: “анонас помочь сегодня вспомнить один вещь который слышать лекция утверждать правдивость хотеть найти источник помнить ещё это услышать общий человек говорить…”

Затем производится балансировка классов с помощью андерсэмплинга по минимальному:



Затем посты кодируются числами с помощью словаря уникальных слов.

Также необходимо выровнять векторы постов по длине. Длина - 50 слов. Для этого длинные посты обрезаются, а короткие - дополняются специальными пустыми словами.

В конце данные преобразуются из списка в np.array-и и разбиваются на обучающую, валидационную и тестовую выборки.

```
X_train.shape, y_train.shape
```

```
((26529, 50), (26529,))
```

```
X_val.shape, y_val.shape
```

```
((3411, 50), (3411,))
```

```
X_test.shape, y_test.shape
```

```
((7959, 50), (7959,))
```

4. Архитектура НС

Была использована нейронная сеть, имеющая следующую архитектуру:

1. Input
2. Embedding (размером 64)
3. GRU (16 нейронов, *возвращает все последовательности*)
4. Conv1D (4 фильтра по 8 слов с промежутком 2 слова)
5. Flatten
6. Dense (8 нейронов, активация - ReLU)
7. Dropout (0.3)
8. Output(Dense) (5 нейронов, активация - SoftMax)

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 input_2 (InputLayer)         [(None, 50)]              0

 embedding_1 (Embedding)      (None, 50, 64)            5108800

 gru_1 (GRU)                  (None, 50, 16)            3936

 conv1d_1 (Conv1D)            (None, 22, 4)             516

 flatten_1 (Flatten)          (None, 88)                0

 dense_2 (Dense)              (None, 8)                 712

 dropout_1 (Dropout)          (None, 8)                 0

 dense_3 (Dense)              (None, 5)                 45

=================================================================
Total params: 5,114,009
Trainable params: 5,114,009
Non-trainable params: 0
_________________________________________________________________
```

Embedding в NLP – это процесс представления слов в виде числовых векторов, который позволяет нейронным сетям работать с текстовыми данными. Embedding позволяет снизить размерность входных данных и сохранить векторное представление слова в таком формате, который можно использовать для обучения моделей машинного обучения. Этот слой работает с пространственным контекстом словосочетаний. В TensorFlow эмбеддинг может быть создан как часть нейронной сети и обучен вместе с остальными слоями. Для создания embedding-слоя в TensorFlow используется функция tf.keras.layers.Embedding. Эта функция создает матрицу, в которой каждому слову в словаре соответствует вектор из чисел заданной размерности. Размерность вектора задается параметром embedding_dim.

Для обработки последовательностей, которые возвращает GRU используются одномерные свертки.

Затем карты признаков слов расплющиваются в один вектор и поступают на вход последних полносвязных слоев, выполняющих классификацию. Для устранения переобучения между полносвязными слоями использован Dropout.

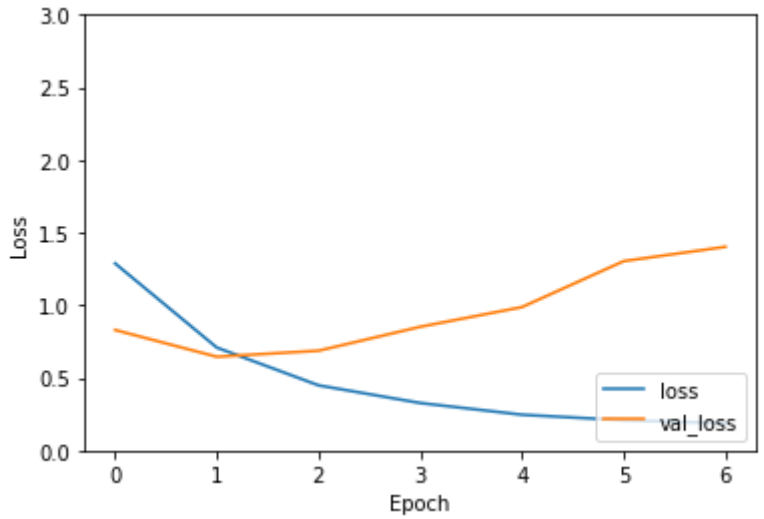
5. Обучение и результаты

Обучение

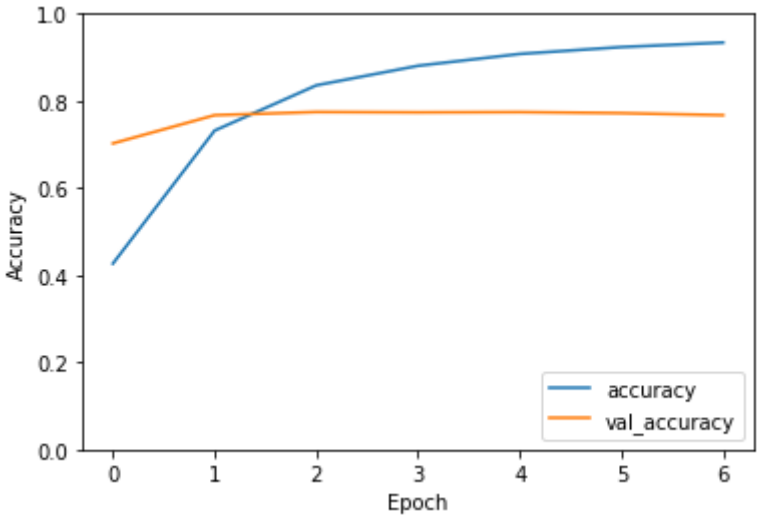
Для обучения нейронной сети были использованы следующие параметры:

1. Loss - SparseCategoricalCrossentropy.
2. Оптимизатор - Adam.
3. BatchSize - 32.
4. 100 эпох.
5. Early Stopping по loss на валидации. “Терпение” - 5 эпох.

Для обучения потребовалось всего 7 эпох. Процесс обучения показан на графике:



Из графика функции потерь видно, что модель переучилась, однако, это каким-то образом не отразилось на снижении accuracy:



Решил оставить так, как есть.

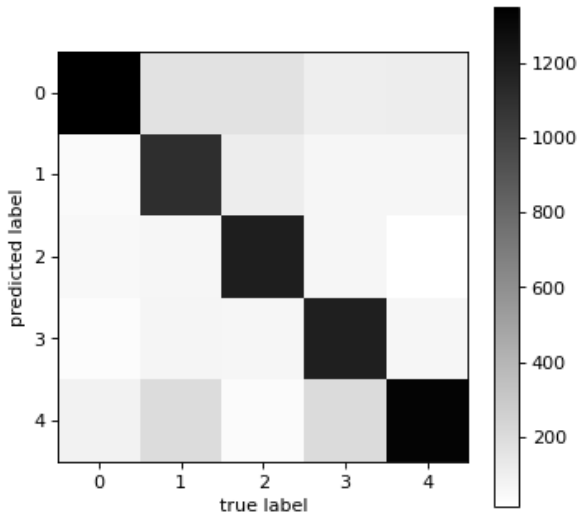
Тестирование

В результате на тестовой выборке были получены следующие результаты:

	precision	recall	f1-score	support
0.0	0.71	0.86	0.78	1566
1.0	0.79	0.69	0.74	1604
2.0	0.86	0.75	0.80	1578
3.0	0.83	0.73	0.78	1625
4.0	0.72	0.84	0.77	1586
accuracy			0.77	7959
macro avg	0.78	0.77	0.77	7959
weighted avg	0.78	0.77	0.77	7959

Точность 0.77 является хорошим результатом при таких “странных” данных. Я удивлен, что тот бред, который я спарсил, имеет какой-то смысл.

Матрица рассогласования:



Диагональ четко выделяется, следовательно, модель действительно хорошо работает.

Примеры классификации (di-столовая, mov-фильмы, ne-животные и природа):

```
это интересный внешке определить впадло ветеринар возить

True label: ne

Predicted label: ne
--------------------------------------------------------------------------------
лор высера камерон понятие вообще несовместимый два терминатор сюжет противоречить
олучать смска будущее

True label: mov

Predicted label: mov
--------------------------------------------------------------------------------
пельмень соус терияк молотый чёрный перец

True label: di

Predicted label: di
```

6. Выводы

В результате работы была решена задача классификации текстов. Был собран датасет, содержащий сообщения из различных разделов 2ch.hk. Из-за ограничений по железу были выбраны пять интересующих классов. Данные были обработаны: удалены лишние слова, преобразованы к начальной форме, удалены слишком короткие сообщения. Проведена балансировка классов. Слова были закодированы. Разработана и обучена нейронная сеть с архитектурой Input-Embedding-GRU-Conv-Dense для классификации сообщений по разделам. Несмотря на переобучение (я пытался устранить...) и весь бред, который пишут на борде, метрики классификации получились довольно неплохими. Модель можно использовать на борде для фильтрации сообщений в общем разделе или как советник раздела при написании сообщения.