УДК 004.89

КЛАССИФИКАЦИЯ ТЕКСТОВ С ПОМОЩЬЮ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

Н. В. Воробьев, Е. В. Пучков
Донской государственный технический университет, Ростов-на-Дону, Российская Федерация
niknegeralmail@gmail.com
puchkoff@i-intellect.ru

Исследованы методы применения сверточных нейронных сетей для классификации коротких текстов — подзадачи разработки чат-бота. Проведено сравнение качества работы построенного классификатора с результатами, полученными с помощью метода опорных векторов и рекуррентных нейронных сетей. Показано, что реализованный классификатор демонстрирует результаты лучше, чем другие методы, а также может быть рекомендован для применения в чат-боте.

Ключевые слова: классификация, обработка естественного языка, сверточные нейронные сети, глубокое обучение.

UDC 004.89

CLASSIFICATION OF TEXTS USING CONVOLUTIONAL NEURAL NETWORKS

N. V. Vorobev, E. V. Puchkov
Don State Technical University, Rostov-on-Don, Russian Federation

niknegeralmail@gmail.com
puchkoff@i-intellect.ru

The article investigates the convolutional neural networks for the classification of short texts as subtasks for the chat-bot development. The constructed classifier performance has been compared with the support vector machine and recurrent neural networks. The implemented classifier has showed better results than other methods and can be recommended for using in chat-bot.

Keywords: classification, natural language processing, convolution neural networks, deep learning.

Введение. Сверточные нейронные сети (*CNN*) успешно зарекомендовали себя в задачах распознавания и классификации изображений. Это привело к множественным попыткам использования этой модели в других областях с целью улучшить результаты в сравнении с традиционными методами. На сегодняшний день классификация текста, т. е. определение принадлежности текста к какой-либо категории (классу) в условиях постоянно возрастающего объема информации является актуальной задачей. Классификация текста применяется в решении многих практических задач, таких как фильтрация документов, распознавание спама, классификация новостей и т. д. Очень часто задача классификации текста является отдельным элементом сложной проблемы, например, создание интеллектуального онлайн-консультанта или чат-бота. В этом случае имеется множество сообщений, разделённых некоторым образом на классы.

Цель данной работы — построить модель (классификатор) с помощью сверточных нейронных сетей для определения типа новых сообщений, которые пользователь посылает чат-боту. Проведение исследований при анализе данных и классификации текста осуществлялось в соответствии с методологией *CRISP-DM* [1]. В рамках поставленной задачи можно выделить следующие этапы:

1. Сбор и подготовка данных для обучения;
2. Построение классификатора;
3. Реализация классификатора в виде программного кода;

4. Обучение и тестирование созданной модели;
5. Верификация модели.

Сбор и подготовка данных. Данные представляют собой короткие сообщения — диалоги, собранные из разных источников: социальные сети; субтитры к фильмам; сообщения, написанные самостоятельно. Количество сообщений приблизительно равно 1700. Было определено 4 класса вышеупомянутых сообщений:

1. беседа;
2. запрос пользователя;
3. контактная информация;
4. информация по доставке.

Сообщения помещены в базу данных. С помощью веб-интерфейса корпус был размечен, т. е. для каждого сообщения был определен класс. Для удобства работы весь корпус переведен в *json*-формат. Проведена предподготовка входных данных. Использовано два подхода — токенизация и технология *Word2Vec*.

Построение классификатора. Сверточные нейронные сети [2], первоначально не предназначались для работы с текстом, они использовались в «компьютерном зрении» и распознавании образов. Сверточная нейронная сеть — это особый вид нейронных сетей прямого распространения (рис. 1). Под прямым распространением понимается то, что распространение сигналов по нейронам идет по порядку, от первого слоя до последнего. Скрытых слоев в сети может быть достаточно много, всё зависит от количества данных и сложности задачи.

Рис. 1. Модель сверточной нейронной сети

Основной особенностью таких сетей является наличие чередующихся слоев типа «свертка — субдискретизация», которых может быть множество. Операция свертки (рис. 2) подразумевает, что каждый фрагмент входа поэлементно умножается на небольшую матрицу весов (ядро), а результат суммируется. Эта сумма является элементом выхода, который называется картой признаков. Взвешенная сумма входов пропускается через функцию активации.

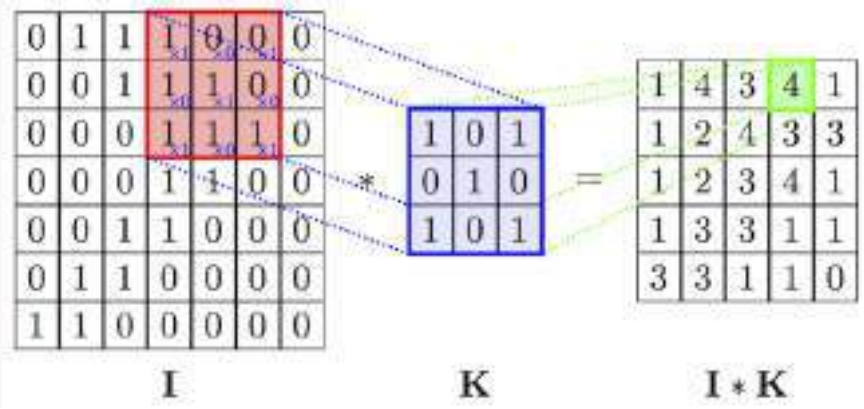

Рис. 2. Слой свертки

Слой пулинга (подвыборки, субдискретизации) представляет собой нелинейное уплотнение карты признаков, проходя нелинейное преобразование. Пулинг интерпретируется как разбиение карты признаков на более мелкие матрицы, нахождение их максимальных элементов, т. е. происходит увеличение «глубины» значений.

Для решения задачи была выбрана следующая архитектура сети:

1. входной слой;
2. слой свертки (*convolution layer*);
3. слой субдискретизации (*max pooling*);
4. полносвязный скрытый слой;
5. выходной слой.

Между основных слоев будут добавлены дополнительные слои для защиты от переобучения сети (*dropout*). Принцип их работы заключается в том, что сеть «забывает» некий процент весов (передается параметром в *dropout*). В качестве активации в слое свертки и скрытом слое будет использована выпрямленная линейная функция активации «*Relu*». Для выходного слоя выбрана функция активации «*Softmax*» — один из частных случаев функции сигмоиды, который используется для многоклассовой классификации.

Программная реализация классификатора. Для реализации данной задачи было выбрано следующее программное обеспечение:

1. язык программирования *Python*, на котором создано множество библиотек для работы с данными и нейронными сетями;
2. библиотека «*Keras*» [3], которая позволяет имплементировать *CNN* с помощью «*TensorFlow*» [4] на более высоком программном уровне;
3. среда разработки *Spyder IDE*.

Отрывок кода реализации сверточной нейронной сети представлен на рис. 3.

```
model = Sequential()

#input layer
model.add(Embedding(max_features,
                    embedding_dims,
                    input_length=maxlen))
model.add(Dropout(0.2))

#convlution layer 1D
model.add(Conv1D(filters,
                 kernel_size,
                 padding='valid',
                 activation='relu',
                 strides=1))
# we use max pooling:
model.add(GlobalMaxPooling1D())

#  hidden layer:
model.add(Dense(hidden_dims))
model.add(Dropout(0.2))
model.add(Activation('relu'))

#output layer:
model.add(Dense(3))
model.add(Activation('softmax'))

```

Рис. 3. Программная реализация модели в *Keras*

Обучение и тестирование модели. Для обучения модели были выбраны следующие параметры:

1. Метрика для ошибки: «*categorical crossentropy*». Используется, когда желательна вероятностная интерпретация оценок. Данная метрика измеряет сходство между истинным классом и прогнозируемым.

2. Оптимизатор: «*adam*». Алгоритм сочетает в себе идею накопления движения градиента и идею более слабого обновления весов для типичных признаков.

3. Метрика для оценки точности классификатора: «*F*-мера». Для многоклассовой классификации выделяют два подхода расчета *F*-меры: *micro*-усреднение и *macro*-усреднение. При *micro*-усреднении значения матрицы несоответствий усредняются по всем классам, а затем вычисляется *F*-мера. При *macro*-усреднении сначала вычисляется *F*-мера для каждого класса, а затем результаты усредняются по всем классам.

Все исходные данные были разделены на обучающую (80 %) и тестовую выборку (20 %). Обучение прекращалось, когда ошибка на тестовой выборке начинала расти, а на обучающей продолжала падать (метод ранней остановки). Обучение модели проходило на ноутбуке *Asus K551L* с видеокартой *NVIDIA 840M*. Графический процессор этой видеокарты поддерживает параллельные вычисления (*CUDA*), что сокращает время обучения в несколько раз по сравнению с вычислениями на *CPU* [5]. Одна эпоха длилась 2–3 секунды (рис. 4).

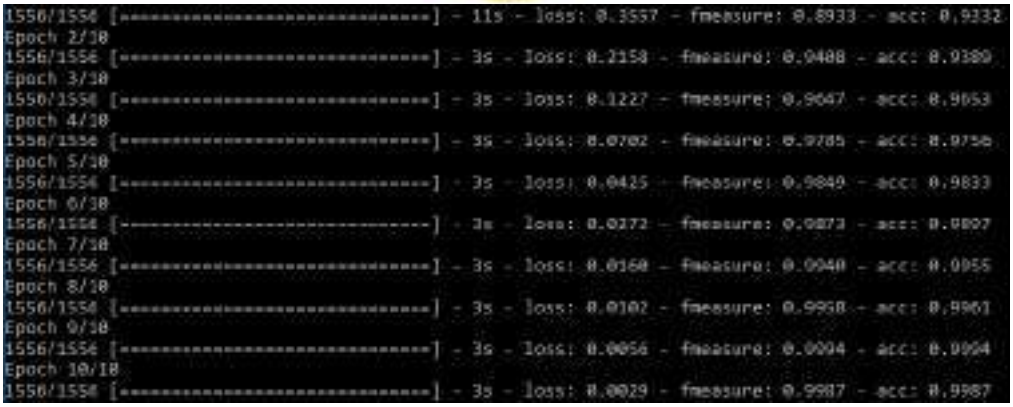

Рис. 4. Обучение модели

Для проведения сравнительно анализа были построены классификаторы с помощью метода опорных векторов [6] и рекуррентных нейронных сетей (*LSTM*) [7]. Результаты тестирования моделей представлены в таблице 1.

Таблица 1

Результаты тестирования

	CNN	*LSTM*	*SVM*
Word2Vec	*Macro*=0,54 *Micro*=0,6	*Macro*=0,48 *Micro*=0,56	*Macro*=0,43 *Micro*=0,51
Tokenization	*Macro*=0,6 *Micro*=0,62	*Macro*=0,5 *Micro*=0,59	*Macro*=0,47 *Micro*=0,55

На основе проведенного тестирования можно сделать следующие выводы: *SVM* и *LSTM* показали результаты хуже, чем *CNN*; подготовка данных с помощью токенизации оказалась эффективней, чем *Word2Vec* из-за небольшой по размеру обучающей выборки. Отметим, что время построения классификатора на основе *LSTM* было больше, чем с помощью *CNN*. Таким образом, для большой выборки данных использование *CNN* будет предпочтительнее.

Верификация модели. Модель *CNN* была проверена на контрольных примерах (размер выборки составил 60 сообщений), неиспользованных при обучении. Для демонстрации результатов создан консольный интерфейс (рис. 5).

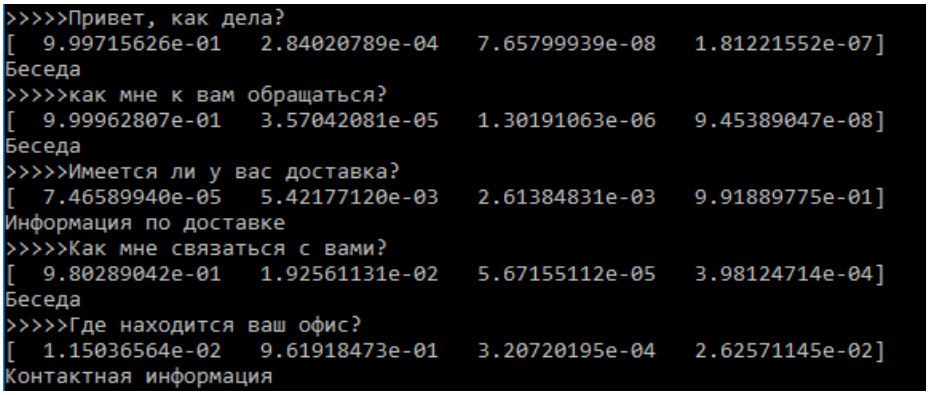

Рис. 5. Результаты проверки модели

Заключение. В данной работе проводилось исследование сверточных нейронных сетей для классификации текстов с целью использования результатов в реализации чат-бота. Проведено

сравнение качества работы построенного классификатора с результатами, полученными с помощью *SVM* и *LSTM*. Доказано, что реализованный классификатор демонстрирует результаты лучше, чем другие методы. Так как создание чат-бота связано с обработкой большой количества текстов, применение сверточных нейронных сетей будет эффективнее с точки зрения вычислительной сложности, чем *LSTM*. Важно отметить, что в данной работе реализована простая архитектура сверточной нейронной сети. Улучшение параметров модели, посимвольная предподготовка данных [8], а также увеличение размера исходного корпуса текстов позволит значительно повысить характеристики классификатора.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 17–01–00888 а.

Библиографический список.

1. IBM SPSS Modeler CRISP-DM Guide [Электронный ресурс] / IBM Support. — Режим доступа : fttp: // public.dhe.ibm.com /software /analytics /spss /documentation/modeler/14.2/ en/CRISP_DM.pdf (дата обращения : 23.04.17).

2. Yann Le Cun, Léon Bottou, Yoshua Bengio and Patrick Haffner: Gradient Based Learning Applied to Document Recognition, Proceedings of IEEE, 86(11):p.2278-2324, 1998.

3. Keras: The Python Deep Learning library [Электронный ресурс] / Официальный сайт библиотеки «Keras». — Режим доступа : https://www.keras.io/ (дата обращения : 23.04.17).

4. About TensorFlow [Электронный ресурс] / Официальный сайт фреймворка «Tensorflow». — Режим доступа : https://www.tensorflow.org/ (дата обращения : 23.04.17).

5. Параллельные вычисления от NVIDIA [Электронный ресурс] / NVIDIA. — Режим доступа : https://www.nvidia.ru/object/cuda-parallel-computing-ru.html (дата обращения : 23.04.17).

6. Corinna Cortes and Vladimir N. Vapnik. Support vector networks. Machine Learning, 20, 1995, pp. 1–25.

7. S. Hochreiter, J. Schmidhuber Long short-term memory // Neural Computation, 1997, 9 (8), pp. 1735–1780.

8. Y. LeCun, X. Zhang. Text understanding from scratch // Computer Science Department, arXiv:1509.01626, 2016.