Московский государственный университет имени М. В. Ломоносова Факультет Вычислительной Математики и Кибернетики

Кафедра ММП, Глубокое обучение

Эссе по теме «Анализ текстов»

Выполнил:
студент 3 курса 317 группы
Зотов Арсений Андреевич

Москва, 2020

Аннотация

В данном эссе рассматриваются задачи интеллектуальной обработки текстов и нейросетевые подходы к их решению. Описываются проблемы, с которыми приходится сталкиваться при обработке текстов. Приводятся примеры некоторых датасетов связанных с обработкой естественного языка и их описания. Работа направлена на ознакомление с наиболее популярными архитектурами сверточных и рекуррентных нейронных сетей, а также их обобщений для анализа текстов. Помимо обзора «классических моделей», приводятся результаты из последних работ по обработке текстов и затрагивается тема механизма внимания для повышения качества предсказания алгоритмов.

Содержание

1 Введение

Технологии, основанные на NLP, получают все большее распространение. Например, телефоны и карманные компьютеры поддерживают интеллектуальное распознавание текста и рукописного ввода; поисковые системы предоставляют доступ к информации, содержащейся в неструктурированном тексте; машинный перевод позволяет нам находить тексты, написанные на китайском языке, и читать их на испанском языке. Языковая обработка стала играть центральную роль в многоязычном информационном обществе. За последние несколько лет нейросетевые методы показали очень хороший результат на различных задачах обработки естественного языка. На практике основными подходами в нейросетях для анализа текстов являются: применение рекурретных и сверточный сетей, использование модели seq2seq и механизма внимания. В данном работе будет приведен обзор популярных архитектур, а также затронуты результаты из последних работ по обработке текстов.

2 Области исследований

Термины:

Токен - элемент последовательности (слово, несколько букв).

Словарь – множество допустимых токенов, то есть модель может принимать/генерировать только их.

3 основных аббревиатуры в анализе текстов [17]:

•NLP(Natural language processing) – всё, что связано с обработкой текстов. NLP стремится преобразовать неструктурированные языковые данные в формат структурированных данных, чтобы машины могли понимать текст и формулировать соответствующие контекстные ответы. На высоком уровне NLU и NLG - это просто компоненты NLP.

•NLU(Natural language understanding) – всё, что связано с пониманием текстов (текст $\rightarrow$ действие / ...). NLU фокусируется на понимании машинного чтения через грамматику и контекст, что позволяет ему определять предполагаемое значение предложения.

•NLG(Natural language generation) – всё что связано с генерацией текста (... → текст). NLG фокусируется на генерации текста или построении текста на английском или других языках с помощью машины и на основе заданного набора данных.

Различают несколько основных типов задач, связанных с текстами. В табл. 1 приведена классификация наиболее популярных задач анализа текстов по виду функциональной зависимости входных и выходных данных.

Тип задачи	**Примеры**
Текст → метка (Классификация)	• определение темы / настроения / автора • определение тональности
Текст → метки (Тегирование)	• определение тегов • разметка на части речи
Текст → текст (seq2seq)	• машинный перевод • аннотирование • чат-бот • продолжение текста • генерация по контенту
Текст, текст → текст	• ответы на вопросы • справочная / экспертная система
... → текст	• описание изображения • моделирование / генерация языка
Текст → ...	• parse tree по предложению • генерация объектов по описанию

Таблица 1: Основные задачи анализа текстов.

Все перечисленные задачи с тесктами, кроме последней, фактически являются классификацией. Классов может быть очень много(число всех токенов).

3 Проблемы

При работе с текстами возникают следующие проблемы:

1. Текст - это сигнал с разделителями.

   В классическом машинном обучении мы привыкли работать с признаковыми описаниями, где каждый элемент является какой-то важной информацией, которую мы не можем заменить. Если же мы поменяем несколько символов в тексте, то мы все равно будем понимать, о чем идет речь(вместо одного пробела напишем два).

2. Один и тот же смысл передается по-разному / синонимы.

   Одну эмоцию можно передать кучей способов, используя синонимы, жаргоны или коверканье языка.

3. Многозначность / омонимы.

   Одно предложение можно интерпретировать по-разному, в зависимости от контекста.

   Пример: "Эти типы стали есть в цехе". Речь идет либо о людях едящих в цехе, либо о присутствии металлических изделий.

4. Динамичность языка, новые слова, сленг.

   Примеры: е-комерция, айпадный, зафрендить и т.п.

5. Устойчивые выражения, профессиональный сленг, контекст.

   Например, выражение "бабье лето"мы не интерпретируем дословно(это не лето и не имеет отношение к женщинам).

   В предложении "Петя увидел Васю, неудивительно, он был очень зоркий/заметный в зависимости от прилагательного мы понимаем, о ком идет речь во второй части предложения.

6. Разметка, как правило, ручная.

7. При попытках признаковой постановки большие разреженные пространства.

8. Текст больше, чем последовательность предложений.

   Важны контекст, порядок изложения, расстановка акцентов и т.д.

3.1 Понимание языка

Важной составляющей в решении задачи обработки текстов является механизм понимания (Language Understanding), который направлен на то, чтобы модель могла автоматически определить намерения пользователя и выдавать желаемый для него ответ. Так, для вопросно-ответной системы (QA) механизм понимания заключается в том, что реализованный алгоритм способен выдавать осмысленные ответы. Например, на вопрос «Когда ходят в школу?» желаемыми ответами являются «в детстве, с сентября», а нежелаемыми – «никогда, вчера».

Другой пример: если изображение представляет собой фото бананов, то ответ «желтые объекты», будет, конечно, верным, но нам бы хотелось получить более конкретный ответ, например, «связка бананов» или «фрукты».

4 История NLP

В истории развития NLP можно выделить следующие основные понятия:

- предобработка (регулярки, стемминг, лемматизация)
- мешок слов (нормировки), N-граммные модели
- языковые модели
- трансферное обучение (перенос обучения)
- мультиязычность
- мультимодальность
- графы знаний

5 Данные.

Существует множество открытых датасетов для работы с текстами.

Несколько примеров:

LAMBADA - предсказать последнее слово, используя, по крайней мере, 50 слов контекста [21].

Children's Book Test - угадать, какое слово пропущено – выбор из 10 [22].

Маленькие датасеты: **Penn Treebank, WikiText-2**

Другие: **The Winograd Schema challenge[23], Conversation Question Answering dataset[24], One Billion Word Benchmark[25], Summarization: CNN and Daily Mail dataset, Translation: WMT-14 English-French[26], Question Answering: Natural Questions dataset[27], GLUE[28], decaNLP[29]**

Многие типичные датасеты устроены следуюшим образом: на вход даётся два текста, первый - некоторая статья, второй - вопрос, ответом на который является искомый выход. В таблице 2 показаны примеры таких датасетов.

В первой версии SQuAD был недостаток: ответы на все вопросы были в пределах параграфа. На смену ему пришел SQuAD 2.0, в котором уже был вариант «нет ответа», что значительно усложнило задачу. В таблице 3 показано сравнение SQuAD 1.1 и SQuAD 2.0.

Dataset	Example	Article / Paragraph
SQuAD	**Q**: How many provinces did the Ottoman empire contain in the 17th century? **A**: 32	**Article**: Ottoman Empire **Paragraph**: ... At the beginning of the 17th century the empire contained 32 provinces and numerous vassal states. Some of these were later absorbed into the Ottoman Empire, while others were granted various types of autonomy during the course of centuries.
CuratedTREC	**Q**: What U.S. state's motto is "Live free or Die"? **A**: New Hampshire	**Article**: Live Free or Die **Paragraph**: "Live Free or Die" is the official motto of the U.S. state of New Hampshire, adopted by the state in 1945. It is possibly the best-known of all state mottos, partly because it conveys an assertive independence historically found in American political philosophy and partly because of its contrast to the milder sentiments found in other state mottos.
WebQuestions	**Q**: What part of the atom did Chadwick discover?† **A**: neutron	**Article**: Atom **Paragraph**: ... The atomic mass of these isotopes varied by integer amounts, called the whole number rule. The explanation for these different isotopes awaited the discovery of the neutron, an uncharged particle with a mass similar to the proton, by the physicist James Chadwick in 1932. ...
WikiMovies	**Q**: Who wrote the film Gigli? **A**: Martin Brest	**Article**: Gigli **Paragraph**: Gigli is a 2003 American romantic comedy film written and directed by Martin Brest and starring Ben Affleck, Jennifer Lopez, Justin Bartha, Al Pacino, Christopher Walken, and Lainie Kazan.

Таблица 2: Пример обучающих данных из датасетов QA [31].

6 Сверточные (CNN) и рекурретные (RNN) нейронные сети для анализа текстов

Входными данными для обучения нейронных сетей являются числовые векторы одинаковой размерности, в то время как слова текста состоят из символов и имеют переменную длину. Таким образом, в обработке текстов возникает необходимость отоб-

	SQuAD 1.1	SQuAD 2.0
Train		
Total examples	87,599	130,319
Negative examples	0	43,498
Total articles	442	442
Articles with negatives	0	285
Development		
Total examples	10,570	11,873
Negative examples	0	5,945
Total articles	48	35
Articles with negatives	0	35
Test		
Total examples	9,533	8,862
Negative examples	0	4,332
Total articles	46	28
Articles with negatives	0	28

Таблица 3: Сравнение SQuAD 1.1 и SQuAD 2.0. [30]

ражения токенов предложения в признаковое пространство размерности k. Одним из наиболее известных способов векторного представления слов (word embedding) является использование инструментов дистрибутивной семантики, таких как Word2Vec, Glove, fastText.

6.1 Сверточные нейронные сети для анализа текстов.

Сверточные нейронные сети успешно показали себя для решения различных задач обработки изображений, почему бы не применить их для задач обработки текстов? Идея использования CNN для анализа текстов возникает ввиду аналогии матричного представления текста и изображения: текст длины n можно представить в виде матрицы $X \in \mathbb{R}^{n \times k}$, каждая строка x_i которой является векторным представлением i-го слова в пространстве признаков размерности k.

Проблема использования сверточных сетей для текстов заключается в том, что в отличие от изображений, тексты представляют собой последовательности произвольной длины. Softmax же определяет вероятности для фиксированного числа классов. В рекурентных сетях данная проблема решается автоматически, так как мы смотрим только на выходы сети.

Впервые архитектура сверточной нейронной сети для классификации текстов была описана в статье [1], схема ее работы приведена на рис. 1.

Для начала каждое слово представляется в виде вектора чисел фиксированной длины. На втором этапе мы применяем свертки. Особенностью данной модели по

сравнению с «классической» CNN для обработки изображений является использование ядра свертки (convolution) фиксированного по ширине размера $h \times k, h \in [1, n]$. Это связано с тем, что лишь целиком каждая строка матричного представления текста является кодом некоторого слова, поэтому нецелесообразно выделять ее фрагменты для обработки. Далее для каждого фильтра после сверточного слоя используем max-pooling. По итогу получаем вектор фиксированной длины, равный количеству сверток. Теперь мы без проблем можем подавать полученный вектор в полносвязную сеть.

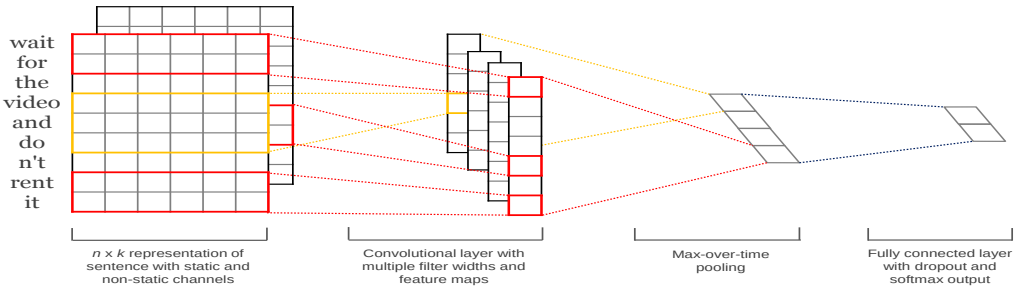

Рис. 1: Архитектура простейшей CNN для обработки текста с одним сверточным и одним полносвязным слоем [1].

Математически описанная архитектура с одним фильтром записывается в следующем виде:

$$c_i = \sigma\left(W \begin{bmatrix} x_i \\ \dots \\ x_{i+h-1} \end{bmatrix}_{hk \times 1} + b\right), i = 0, \dots, n-h,$$

где $W \in \mathbb{R}^{1 \times hk}$ – фильтр свертки (настраивается в процессе обучения),
$h \in \mathbb{N}$ – высота ядра свертки, $b \in \mathbb{R}$ - смещение.

По фильтру W получается вектор $c = [c_0, \dots, c_{n-h}] \in \mathbb{R}^{n-h+1}$, который при помощи метода max-pooling сворачивается в ответ $c' = \max(c) \in \mathbb{R}$.

При использовании нескольких фильтров полученные значения c' по каждому фильтру конкатенируются, далее для получения ответа к ним применяется полносвязный слой.

Идеи для улучшения работы сверточных моделей для текста:

- применение сверток разной высоты для разных фильтров – пример использования в задаче классификации текстов описан в статье [2]

• динамические сверточные нейросети (Dynamic Convolutional Neural Network) [3]:

✓ k-pooling вместо max-pooling – возвращает k наибольших элементов вектора в порядке убывания, по которым с помощью добавления нелинейности строятся признаковые карты (feature map). Данная операция позволяет терять меньше информации при пулинге.

✓ динамический k-pooling – параметр k зависит от параметров сети (глубины сети, длины входа). Для нас важна только фиксированная длина конечного вектора.

✓ широкие свертки: недостающие элементы окна заполняются фиктивно. Принцип работы узкой и широкой свертки изображен на рис. 2.

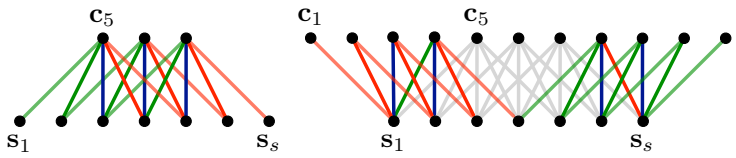

Рис. 2: Узкая и широкая свертка для фильтра размера 5 [3].

✓ Дополнительно в DCNN используется метод folding, сворачивающий матрицу с данными в 2 раза, например, при помощи суммирования. Архитектура DCNN приведена на рис. 3.

• Multichannel Variable-Size CNN [4] – основной особенностью модели является использованиие мультиканального входа. Алгоритм принимает на вход трехмерный тензор, который состоит из матричных представлений некоторого предложения, полученных при помощи разных подходов к векторному представлению слов: word2vec, GloVe, SENNA и др. В целом, архитектура данной сети похожа на DCNN, рис. 4.

Основные концепеции MVCNN заключаются в следующем:

✓ На каждом сверточном уровне генерируются карты признаков (feature maps). Для этого в случае одноконального входа к входным данным применяется несколько сверток с ядрами одинакового размера, затем их результаты конкатенируются в матрицу – карту признаков. Когда каналов

несколько, они обрабатываются параллельно и карта признаков получается объединением результатов по ним.

✓ Используется динамический k-pooling

✓ Используются широкие свертки

✓ Для получения ответа в задаче классификации используется слой логистической регрессии, получающий на вход результат работы полносвязного слоя после заключительного k-pooling

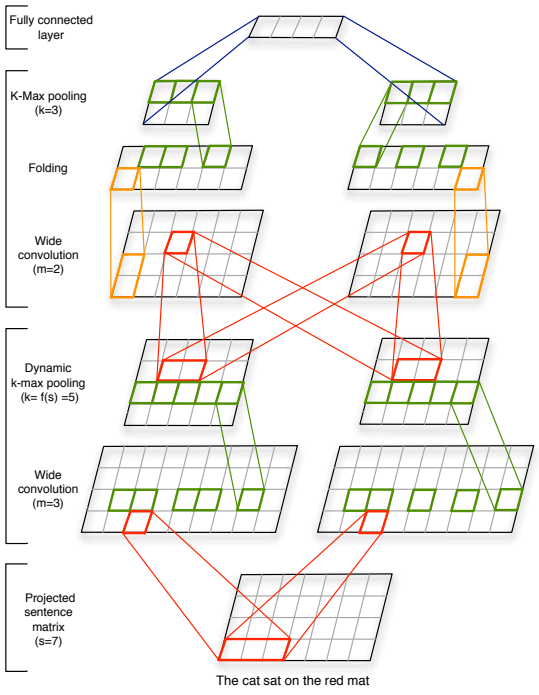

Рис. 3: DCNN с 2 сверточными слоями [3].

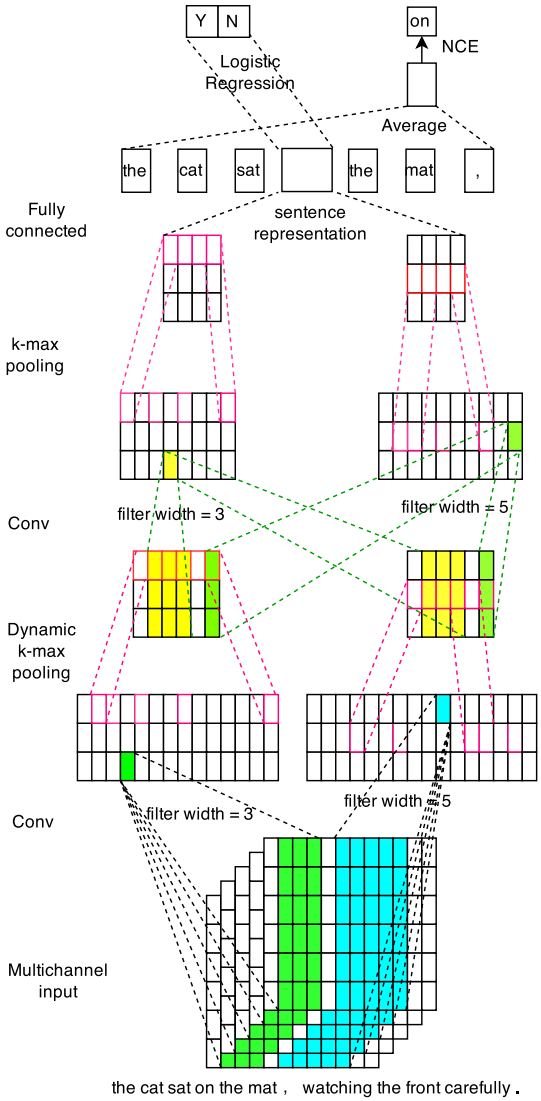

Рис. 4: MVCNN с 2 сверточными слоями [4].

• Very Deep CNN [5] – идея метода заключается в применении глубоких CNN (до 29 сверточных слоев) к посимвольному представлению текста. Свертки на каждом уровне являются небольшими, длины 3, однако в объединении позволяют более точно анализировать целые n-граммы. Архитектура данной сети схожа с известными VGG и ResNet, представлена на рис. 5.

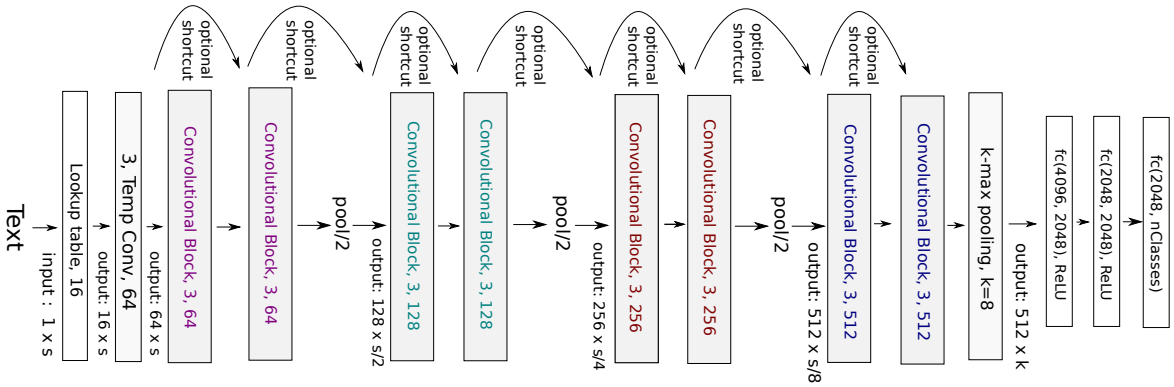

Рис. 5: Архитектура VDCNN [5].

Изображенная модель сети VDCNN содержит один обязательный сверточный блок из 64 сверток размера 3 и нескольких последующих блоков различного размера с использованием shortcut соединения для решения проблемы затухающих градиентов. Сверточные блоки (рис. 6) содержат по 2 сверточных слоя с применением Batch-normalization и результирующей функцией активации ReLU. По аналогии с VGG, разные по количеству сверток группы «convolutional blocks» разделены операцией pooling, сжимающей промежуточные данные в 2 раза с целью экономии памяти. Так же реализована идея Temporal BatchNorm - для минибатча из m обектов на последовательности длины s статистики считаются по m*s слагаемых.

Авторы статьи провели исследования и пришли к следующим выводам:

• Глубина важна. Максимальное качество достигалось на 29 слоях.

• Max-pool оказался лучше всех модификаций.

• Эта модель лучше предыдущих.

• Прокидывание связей помогает, но только на большой глубине.

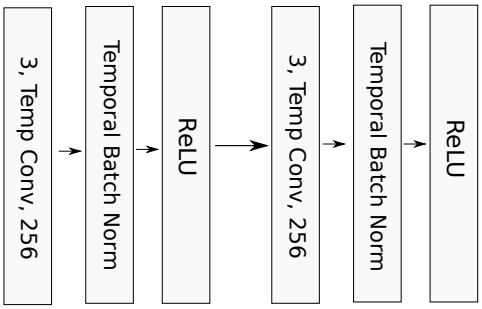

Рис. 6: Сверточный блок VDCNN [5].

6.2. Рекуррентные нейронные сети для анализа текстов

Другой парадигмой алгоритмов для анализа текстов являются рекуррентные нейронные сети (RNN) - подкласс нейронных сетей с обратными связями, которые используют предыдущие состояния сети для вычисления текущего. Для анализа текстов рекуррентные сети принимают на вход упорядоченную последовательность векторизованных слов текста. Использование механизма связывания предыдущей инфмормации с текующей делает рекурретные сети привлекательными для обработки текста, однако на практике проблемой «простых» RNN является «забывание» долговременных зависимостей и затухающие градиенты.

Архитектура LSTM (Long short-term memory) [6] позволяет бороться с этими недостатками, благодаря использованию методов «гейтов», контролирующих запоминание информации из потоков входов и ее учета на выходах внутри базового блока.

Более простым аналогом LSTM является модель GRU (Gated recirrent unit) [7] , которая в отличии от LSTM из базового блока передает только одно состояние. Как и сети LSTM, GRU способны обнаруживать долгосрочные зависимости в данных и бороться с проблемой затухания градиентов.

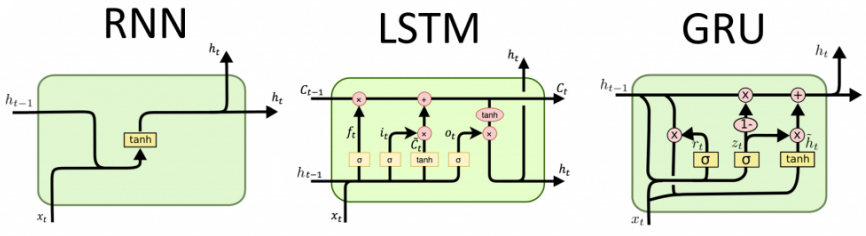

Рис. 7: Схема базового блока для RNN, LSTM, GRU [8].

6.3 Сравнение CNN и RNN

При сравнении качества работы рекуррентных и сверточных сетей на различных типах задач анализа текста, как-то: определение настроения текста, разметка на части речи, выбор ответа - показано, что ни одна сеть из CNN, LSTM, GRU не обладает явным преимуществом во всех задачах. Подробное сравнение в статье [9].

Задача (метрика качества)	**Модель**	**Качество**
Sentiment Classification (acc)	CNN	82.38
	GRU	**86.32**
	LSTM	84.51
Relation Classification (F1)	CNN	68.02
	GRU	**68.56**
	LSTM	66.45
Textual Entailment (acc)	CNN	77.13
	GRU	**78.78**
	LSTM	77.85
Answer Selection (MAP & MRR)	CNN	(**63.69,65.01**)
	GRU	(62.58,63.59)
	LSTM	(62.00,63.26)
Question Relation Match (acc)	CNN	**71.50**
	GRU	69.80
	LSTM	71.44
Path Query Answering (hit@10)	CNN	54.42
	GRU	**55.67**
	LSTM	55.39
Part-of-Speech Tagging (acc)	CNN	94.18
	GRU	93.15
	LSTM	93.18
	Bi-GRU	94.26
	Bi-LSTM	**94.35**

Таблица 4: Результаты моделей CNN, GRU, LSTM в задачах анализа текста [9].

Результаты показывают, что для большинства задач модели RNN выигрывают в точности, кроме задач на сопоставление вопросов и ответов. В целом, оба типа нейросетей показывают сравнимое качество на исследуемых задачах в текстах.

6.4 Совмещение CNN и RNN

C-LSTM

Идея совмещения двух типов нейросетей CNN и RNN в одну модель C-LSTM для повышения качества обработки текстов впервые описана в статье [10].

Архитектура C-LSTM (рис. 8) представляет собой 2 основных блока:

1. **CNN** - служит для высокоуровневого представления текста в векторизованном виде (учитывает связи между словами). На сверточном уровне CNN для получения карт признаков используются свертки одинакового размера. Далее, для генерации последовательности векторов, соответствующие друг другу элементы карт признаков конкатенируются в векторы. Операция max-pooling после сверточного уровня не применяется.

2. **LSTM** - служит для построения итогового предсказания на основе улучшенного представления текста из CNN

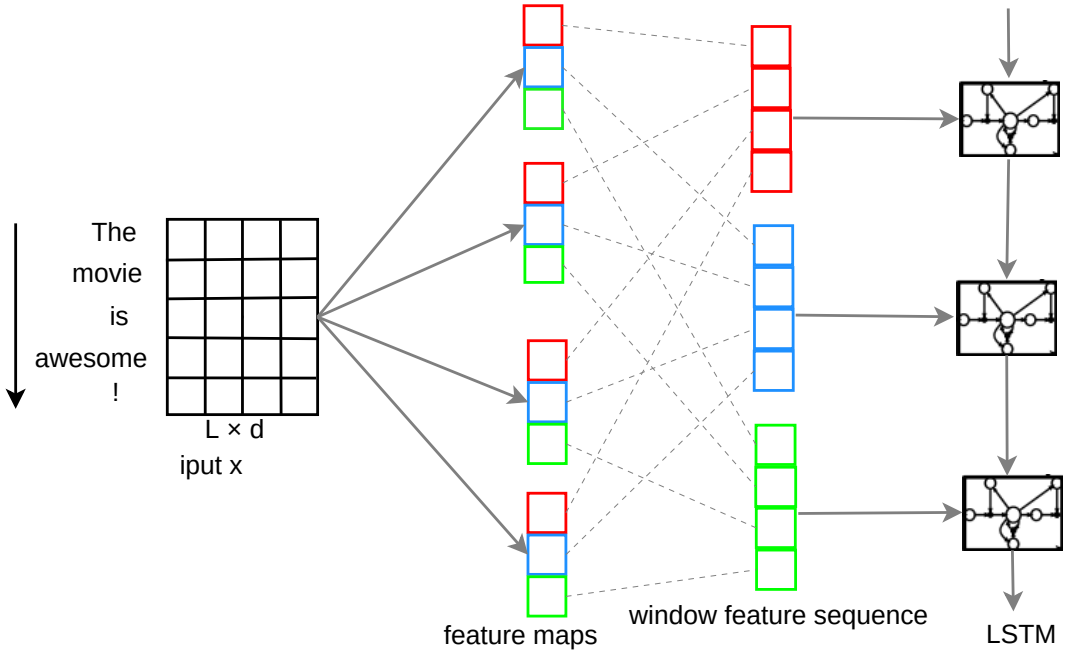

Рис. 8: Архитектура C-LSTM [10].

LSTM-CNNs-CRF[18]

Можно конкатенировать и большее количество архитектур. Примером служит LSTM-CNNs-CRF, являющаяся конкатенацией CNN, LSTM и CRF [10].

На рис. 9 показана CNN, которая используется для извлечения посимвольного представления слова. До применения сверток применяется dropout-слой.

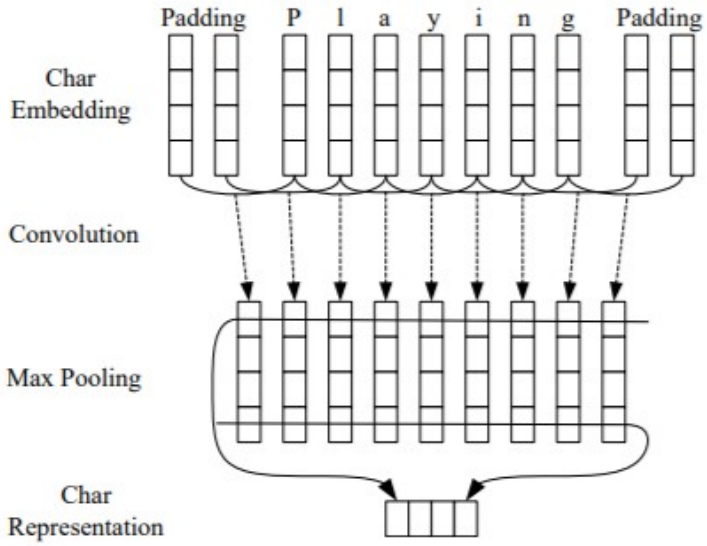

Рис. 9: CNN для посимвольного представления слова. Пунктирные стрелки указывают на dropout-слой [18].

Архитектура LSTM-CNNs-CRF (рис. 10) представляет собой следующие этапы:

Для каждого слова с помощью CNN вычисляется посимвольное представление. Затем вектор посимвольного представления объединяется с embedding-вектором слова для подачи в слой BLSTM. Наконец, выходные векторы BLSTM подаются в слой CRF для получения наилучшей последовательности меток. Стоит заметить, что Dropout-слой применяется как к входному, так и к выходному векторам BLSTM.

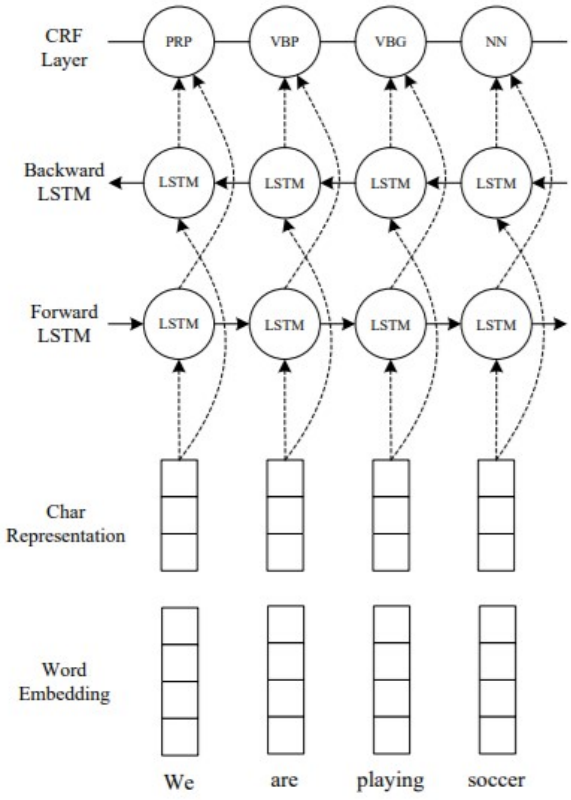

Рис. 10: Архитектура LSTM-CNNs-CRF [18].

7 Модель seq2seq

Модель sequence to sequence [11] является одним из наиболее популярных алгоритмов в задачах машинного перервода и вопрос-ответных системах.

Архитектура seq2seq (рис. 11) представляет собой объединение 2 рекурретных сетей:

1. **кодировщик (encoder)** - служит для векторного представления входного текста. Кодировщик принимает на вход последовательность векторизованных слов исходного текста, а на выходе выдает вектор контекста c фиксированной длины.

2. **декодировщик (decoder)** - служит для построения целевой последовательности слов (перевода текста на другой язык или ответа на вопрос). На шаге t декодировщик предсказывает слово y_t по вектору контекста c, состоя-

нию сети h_t, и предыдущим значениям $y_1, \ldots, y_{t-1}$. На этапе обучения вместо сгенерированных y_i используются целевые слова.

Значение y_t определяется как наиболее вероятное слово с точки зрения условного распределения на всем словаре: $p(y_t|y_1, \ldots, y_{t-1}, c) = g(y_{t-1}, s_t, c)$, s_t – скрытое состояние. На первой итерации в качестве y_0 подается стартовое слово EOS, далее процесс построения ответа продолжается, пока не будет сгенерирован маркер конца строки EOL.

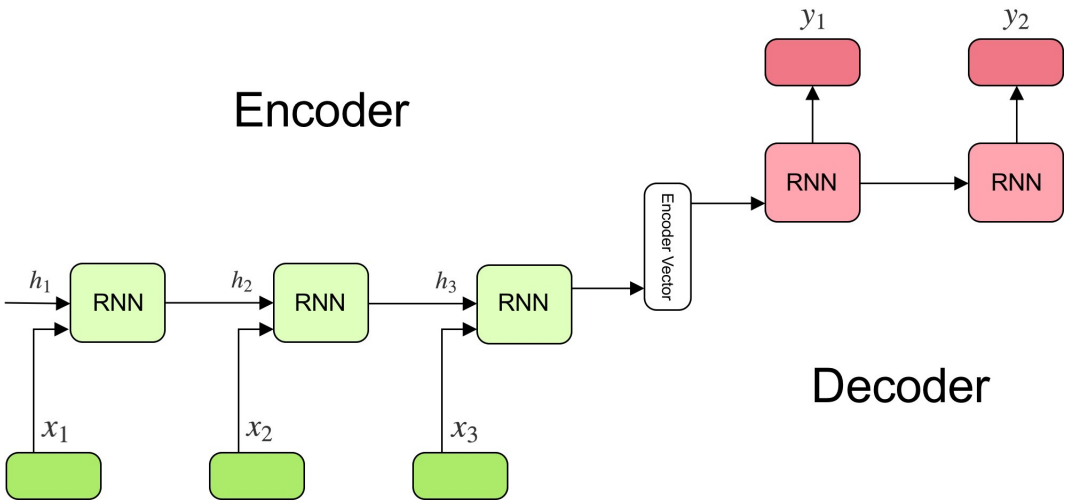

Рис. 11: Архитектура seq2seq [12].

Недостатком описанной модели является тот факт, что на каждом шаге декодер жадным способом выбирает наиболее вероятное слово в текущей позиции, однако такой подход не гарантирует генерацию наиболее вероятного предложения в рамках модели.

Для решения данной проблемы предложен алгоритм beam-search [13]. Его идея заключается в запоминании нескольких наиболее вероятных слов на каждом шаге и выборе в качестве результата наиболее вероятной порожденной последовательности с учетом всего контекста.

Недостатком архитектуры seq2seq является то, что всё предложение представляется 1000-мерным вектором. Вряд ли этого будет достаточно для представления всех предложений языка и для качественного перевода. На помощь приходит механизм внимания.

Эмпирическим путем было выявлено, что в задаче машинного перевода качество предсказания повышается при инвертировании порядка слов входной последовательности.

В простейшей реализации в декодировщике передается только его внутреннее состояние. Выход кодировщика передается лишь первому элементу. Вместо этого можно передавать его на каждом шаге декодировщика, чтобы информация не утрачивалась.

Если хорошо обучить модель и визуализировать вектора(рис. 12), полученные с помощью декодировщика, то получим что вектора будут иметь осмысленное расположение в пространстве.

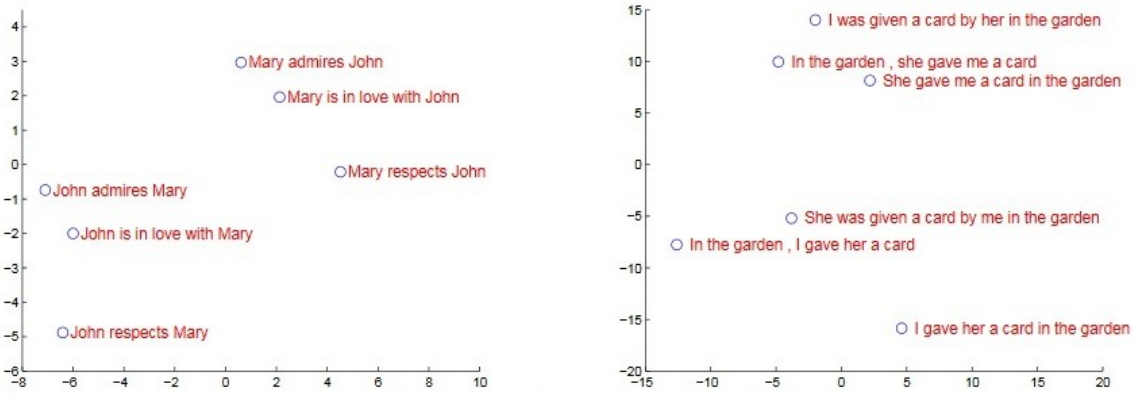

Рис. 12: Внутреннее представление предложений [11].

8 CANINE [20]

Все часто используемые модели по-прежнему требуют явного шага токенизации. Хотя недавние подходы к токенизации, основанные на лексиконах подслов, полученных из данных, менее хрупки, чем токенизаторы, созданные вручную, эти методы не подходят одинаково для всех языков, и использование любого фиксированного словаря может ограничить способность модели к адаптации.

В работе пердставлен CANINE - encoder, который работает непосредственно с последовательностями символов - без явной токенизации или словарного запаса. Входными данными являются последовательности символов Юникода. Чтобы представить полное пространство символов Юникода без словаря, используется стратегия

хеширования. Чтобы избежать замедления из-за увеличения длины последовательности модели, входные данные пропускаются через свертки со сдвигом.

Архитектура CANINE (рис. 13) представляет собой следующие этапы:

1. **Предобработка**: входные данные для CANINE должны быть представлены как последовательность целых чисел, но поскольку представление символов четко определено и стандартизировано Unicode, можно просто брать их целочисленные значения в Unicode.

2. **Получение представлений с помощью хэша(Hash Embedding)**: CANINE получает представление целочисленных значений Unicode, используя несколько хеш-функций. Это позволяет модели представлять все 143 тыс. символов Unicode с относительно небольшим количеством параметров.

3. **Понижение размерности(Downsampling)**: чтобы CANINE была эффективной, используется стратегию понижения размерности с несколькими этапами. Сначала символы кодируются с помощью однослойного local-трансформера. Затем применяется свертка со сдвигом, чтобы уменьшить количество позиций в последовательности.

4. **Применение трансформера(Deep transformer stack)**: после понижения размерности применяется глубокий трансформер с L слоями. Этот этап считается ядром CANINE, поскольку на него приходится подавляющее большинство его вычислений и параметров. Эту часть модели можно легко заменить любой другой seq2seq моделью.

5. **Повышение размерности(Upsampling)**: хотя описанной выше архитектуры достаточно для задач классификации, для задач прогнозирования последовательности требуется, чтобы модель предоставляла выходной слой с той же длиной последовательности, что и входной. Выходное представление реконструируется на уровне символов, сначала объединяются выходные данные полученные на предыдущем этапе с представлением, полученным при понижении размерности. Затем примененяется последний слой трансформера (стандартного, а не local), с с помощью которого получаем окончательное представление последовательности.

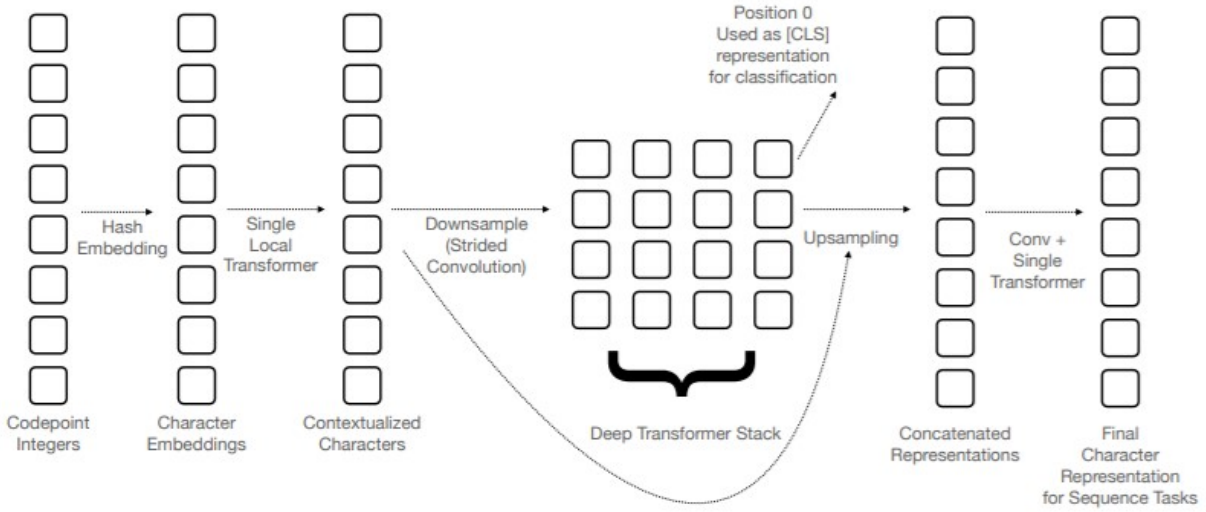

Рис. 13: Архитектура Canine [20].

В той же статье продемонстрировано сравнение данного подхода с другим популярным подходом кодирования mBERT, где CANINE превосходит mBERT в задаче TYDI QA SELECTP. Применение идеи понижения размерности приводит к значительному увеличению скорости работы алгоритма, а так же к небольшому приросту качества. При этом используется меньшее количество параметров.

9 Механизм внимания (attention) в обработке текстов

Attention [14] — это концпеция, заключающаяся в выделении части входных данных (фрагментов текста), на которые при решениии некоторой задачи стоит обратить больше внимания. Например, в задаче машинного перевода необходимо учитывать взаимосвязи между словами, чтобы целевое предложение было лексически и грамматически корректным.

Мотивация применения механизма внимания отображена на рис. 14. Очевидной концепцией языка является то, что между словами есть взаимосвязи. Интуитивно понятно, что в рамках одного предложения слова green, apple и eating, apple сильно связаны, а eating, green - слабо; метод attention направлен на автоматическое извлечение подобных связей.

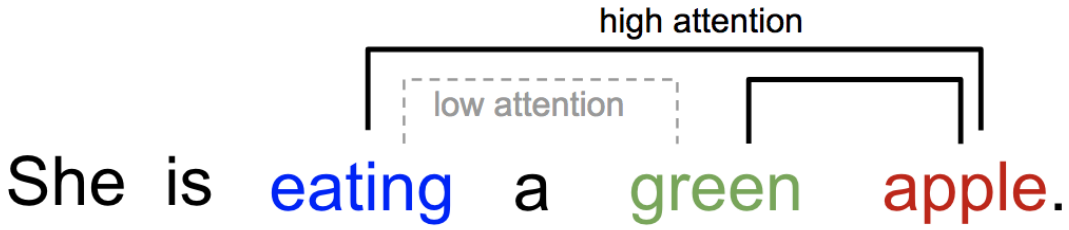

Рис. 14: Иллюстрация механизма внимания для анализа предложения [14].

Добавление механизма внимания в архитектуру seq2seq [15] позволяет качественно решить проблему сжатия входного текста в единственный вектор контекста c: в обобщенной модели для каждой позиции целевого предложения вектор контекста будет вычислен на основе всех токенов исходного текста с учетом внимания.

Внимание описывается в виде вектора весов α: для каждого шага j декодировщика вектор $\alpha_i \in \mathbb{R}^n$ содержит информацию о важности исходных токенов $x_1, \ldots, x_n$ для определения целевого слова в позиции t. На рис. 15 представлена схема работы обобщенного алгоритма seq2seq с механизмом внимания [12].

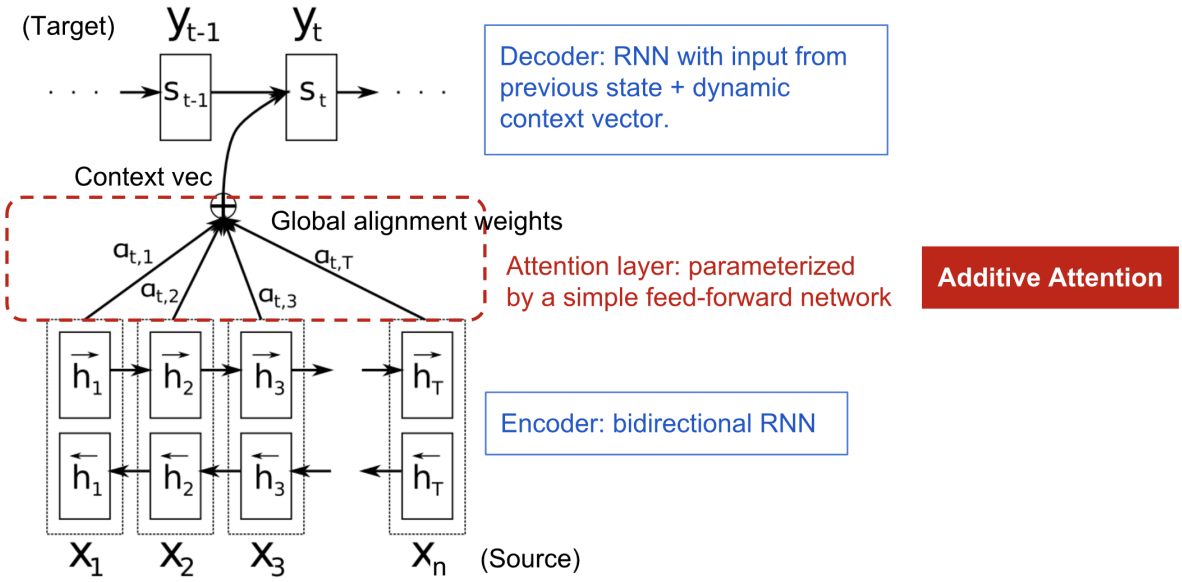

Рис. 15: Обобщение модели seq2seq для BiRNN с механизмом внимания [14].

Для реализаци механизма внимания предложен следующий подход:

1. $h_j = \left[\overrightarrow{h}_j^\top; \overleftarrow{h}_j^\top\right]^\top$ – вектор состояния кодировщика на шаге j

2. $c_i = \sum_{j=1} \alpha_{ij} h_j$ – вектор контекста на шаге i декодировщика

3. $\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$ – нормированный вес

4. $e_{ij} = a(s_{i-1}, h_j)$ – мера соответствия (alignment) состояний h_j кодировщка и s_{i-1} декодировщика.

   Различные способы задания функционала выравнивания a:

   (a) $a(s, h) = s^T h$ – скалярное произведение как мера похожести

   (b) $a(s, h) = s^T W h$, где W – настраиваемая матрица параметров

   (c) $a(s, h) = w^t tanh(W_1 s - W_2 h)$, где w, W_1, W_2 – настраиваемые матрицы параметров

   (d) $a(s, h) = s^T h / \sqrt{d}$ – масштабированное скалярное произведение, d - размерность векторов

   (e) $a(s, h) = cos(s, h)$ – косинусное расстояние

Использование механизма внимания позволяет визуализировать и интерпретировать процесс машинного перевода через матрицы выравнивания – матрицы, в которых строке i соответствует i-ое слово исходного предложения, а каждому столбцу j - j-ое слово целевого. Элемент матрицы в позиции (i, j) отображает вес α_{ij}: черный цвет соответствует $\alpha_{ij} = 0$, белый - $\alpha_{ij} = 1$. Матрица выравнивания на рис. 16 показывает, что для большинства слов выравнивание линейно, однако есть и участок с инвертированным порядком.

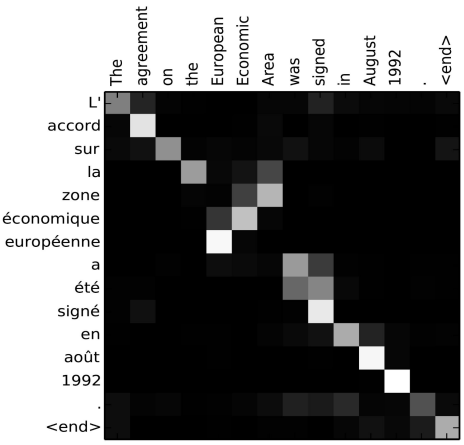

Рис. 16: Матрица выравнивания для перевода предложения с английского языка на французский [14].

Таким образом, механизм внимания (attention) является важным улучшением над моделью seq2seq по целому ряду причин:

- улучшает качество перевода
- решает проблему «узкого горла» вектора контекста
- появляется интерпретируемость
- решает проблему «затухания сигнала» / исчезающего градиента
- получаем выравнивание (alignment) «бесплатно» в переводе

Виды внимания

В машинном обучении принято выделять 3 основных типа внимания:

1. **Self-Attention / intra-attention** [16] – направлено на разные позиции одной и той же входной последовательности, рис. 17
2. **Global / Soft attention** – направлено на весь вход; модель дифференцируема, но требует много вычислений, рис. 18
3. **Local / Hard attention** – направлено на некоторую часть входа с целью сокращения вычислительных затрат; модель становится недифференцируемой, рис. 18

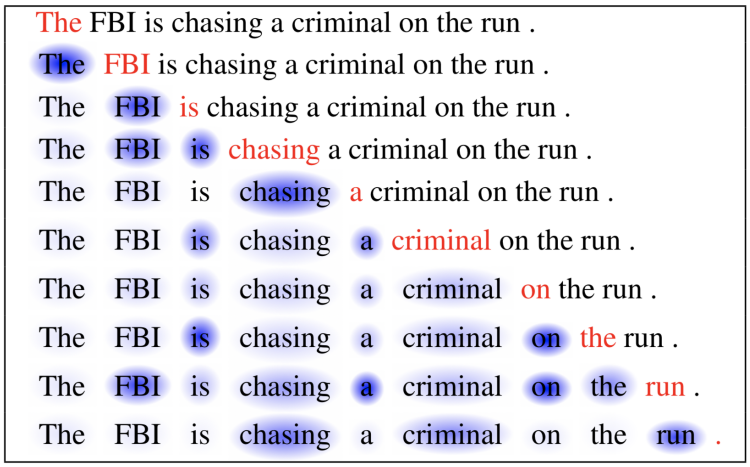

Рис. 17: Иллюстрация Self-Attention для нейросети при чтении предложения. На каждой итерации текущее слово выделено красным цветом, а слова, на которые сеть обращает внимание, – синим [14].

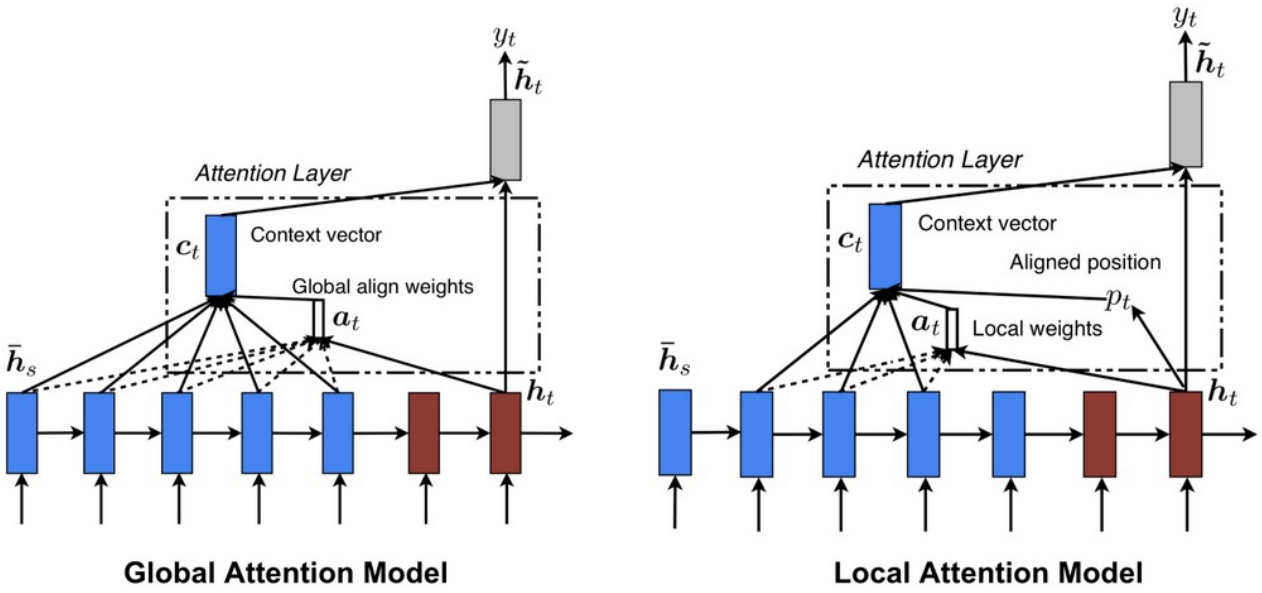

Рис. 18: Различие Global и Local attention [14].

10 Exclusion and Inclusion [19]

Глубокое обучение продемонстрировало превосходную производительность по сравнению с традиционными методами машинного обучения. Это связано с их способностью находить сложные нелинейные зависимости. Однако неспособность эффективно объяснить эти зависимости привела к тому, что нейронные сети были охарактеризованы как черные ящики. Более того, использование моделей черного ящика подверглось критике из-за отсутствия объективности и интерпретируемости. Для обработки естественного языка, особенно в задачах seq2seq, существует относительно немного методов, которые могут извлекать зависимости, которым обучился DNN.

В работе представлен метод "исключения-включения"для фразовой важности признаков в DNN. Данный метод работает как для задач классификации, так и задач регрессии:

• Для задач классификации выделяется, какие фразы относятся к определенному классу.

• Для задач регрессии мы рассчитывается положительное / отрицательное влияние фраз на общую оценку.

Постановка задачи.

Нам дан набор данных $D = \{x_{ki}, y_{ki}\}_{i=0}^{N}$. Каждый x_{ki} представляет собой строку текста, а метка $y_i \in \{1....q\}$ указывает метку / значение x_{ki} среди заданного набора из q меток. Подбираем модель F такую, что $y_{ki} = F(x_{ki})$.

Вычисление значимости (Importance Calculation).

Этот шаг очень важен при расчете значимости для регрессии, потому что после удаления неважных слов мы вычисляем влияние на результат только важных фраз. Мы не рассчитываем эффект любого слова (или фразы), которое не имеет значения. В модели глубокого обучения NLP мы передаем входные данные как последовательность, и появление одного и того же слова (или фразы) в одном и том же месте в обучающих данных почти не повторяется. Это означает, что количество расположений этого слова (или фразы) в определенном месте почти всегда равно 1, что затрудняет вычисление стандартной ошибки этого слова (или фразы) в этом месте.

Чтобы решить эту проблему, авторы работы использовали функцию потерь задачи в качестве основы и, проверяя изменение этой функции потерь, решали, важно это слово (или фраза) или нет. Для примера регрессии вычислим прогнозируемый ответ, исключив и включив одну фразу, скажем, $phr_i = [w1, w2, .., wn]$, и получим на выходе соответственно y_{ex} и y_{in}. Вычислим функцию потерь(например, MSE) для каждого из выходов. Теперь, если $MSE_in < MSE_ex$, то включение phr_i снижает потери, иначе фраза не важна. После удаления всех неважных фраз в соответствии с описанной процедурой, если прогнозировать вывод только по важным словам, то новый вывод становится более точным по сравнению с предыдущим выводом, полученным на основе всех слов. Т.е $MSE_{imp} <= MSE_{all}$.

Та же процедура не применяется для классификации, потому что для классификации вместо прямой оценки мы прогнозируем вероятность того, что данные будут отнесены к классу. Для классификации мы вычисляем, какие слова (или фразы) увеличивают или уменьшают вероятность попадания данных в этот конкретный класс. Это было учтено при вычислении влияния.

Вычисление влияния (Effect Calculation).

Влияние указывает на положительное или отрицательное влияние слова (или фразы) на вывод, то есть для регрессии, если какая-либо фраза phr_j имеет положительное влияние, это значит, что включение этой конкретной фразы увеличит выход y, а исключение - уменьшит его. И наоборот, если эта фраза имеет негативное влияние. Для классификации положительное влияние любого слова (или фразы) означает, что включение этого слова (или фразы) увеличивает вероятность того, что данные отнесутся к определенному классу, и наоборот, для отрицательного влияния.

Перед вычислением влияния неважные слова зануляются. 0 означает отсутствие этого слова, что аналогично тому, что мы делаем во время padding или маскирования. Расчет влияния схож с расчетом значимости, но вместо функции потерь здесь мы берем прогнозируемый результат модели. Для регрессии, допустим, включая фразу $phr_i = [w1, w2, .., wn]$, мы получаем прогнозируемый ответ y_{in}, а исключая эту фразу, мы получаем прогнозируемый ответ y_{ex}. Если $y_{in} > y_{ex}$, это означает, что эта фраза положительно влияет на результат и наоборот, если уравнение обратное. Для классификации, если существует n классов, мы вычисляем вероятности каждого класса, включая интересующую фразу, и вероятности, исключая ее. Скажем, вероятности класса c_j, включая и исключая phr_i, равны $P_{in}(c_j)$ и $P_{ex}(c_j)$ соответственно. Если $P_{in}(c_j) > P_{ex}(c_j)$, это означает, что включение этой фразы увеличивает вероятность того, что точка данных попадет в класс c_j, и наоборот.

Метрика оценки.

Метрика оценки влияния - это процентное изменение выходных данных модели относительно исключения и включения фраз. Авторы называют ее EI (Exclusion-Inclusion) score:

$$EI(phr_i) = \frac{y_{in} - y_{ex}}{y_{in}} * 100$$

Если оценка EI положительна, фраза имеет положительный эффект, а если отрицательная, то она отрицательно влияет на результат. Вышеприведенное уравнение верно для регрессии, для классификации y можно просто заменить на $P(c_j)$, чтобы получить влияние фраз на этот конкретный класс.

Алгоритм.

По итогу получаем следующий алгоритм, состоящих из двух этапов:

1. **Вычисление значимости (Только для задачи регрессии).**

   Отфильтруем все важные индексы слов и заменим остальные индексы на 0. Получим прогнозы модели для важных слов, то есть y_{imp}

2. **Вычисление влияния.**

   Фраза с положительной и отрицательной оценкой EI будет иметь положительный и отрицательный эффект на результат соответственно. Неважные фразы (или слова) считаются нейтральными.

Результаты. Далее представлены результаты работы алгоритма для задачи классификации (рис. 19) и задачи регрессии (рис. 20). Красным помечены фразы, которые алгоритм посчитал отрицательными, зеленым - положительными. Остальные фразы являются нейтральными.

Sentence	Predicted Class	Label
a bad movie that happened to good actors	0	0
it's simply stupid irrelevant and deeply truly bottomlessly cynical	0	0
there s no art here it s still a good yarn which is nothing to sneeze at these days	0	1
a fun ride	1	1
overcomes its visual hideousness with a sharp script and strong performances	1	1

Рис. 19: Пример работы Exclusion and Inclusion на задаче классификации [19].

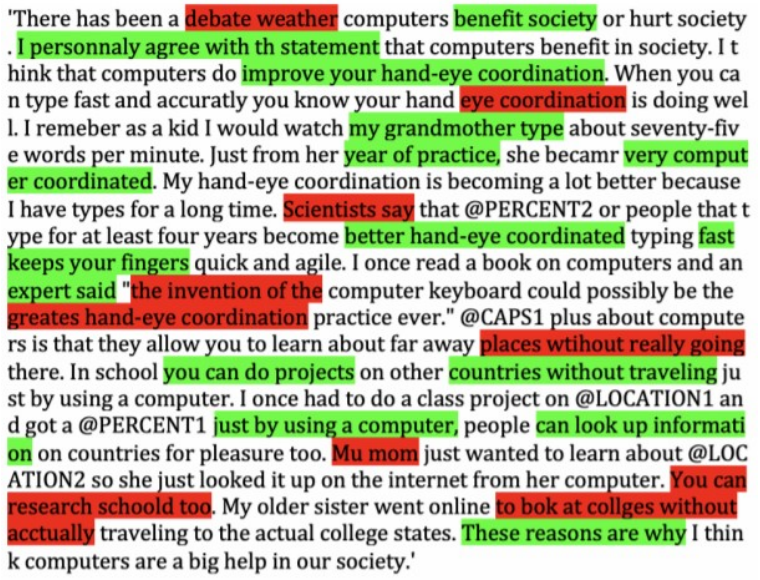

Рис. 20: Пример работы Exclusion and Inclusion на задаче регресии [19].

11 Заключение

Приведенный в работе обзор наиболее известных методов интеллектуальной обработки текстов показал, что сверточные сети могут применяться не только для изображений, но и для обработки текстов. Были представлены рекурентные сети, а так же возможные комбинации CNN и RNN. Познакомились с простейшей и понятной архитектурой seq2seq, а так же ее обобщениями и концепцией внимания.

Упомянутая в работе концепция Exclusion and Inclusion позволила сделать шаг к лучшему пониманию того, как DNN предсказывают текст, в отличие от обработки сложных нелинейных моделей как черных ящиков.

Так же описана концепция CANINE, которая является кодировщиком для понимания языка, который использует модель без токенизации и словаря, превосходя по качеству модели, построенные на основе эвристических токенизаторов.

Список литературы

[1] Yoon Kim, «Convolutional Neural Networks for Sentence Classification,» arXiv preprint arXiv:1408.5882, 2014

[2] Ye Zhang, Byron Wallace, «A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification,» arXiv preprint arXiv:1510.03820, 2016

[3] Nal Kalchbrenner, Edward Grefenstette, Phil Blunsom, «A Convolutional Neural Network for Modelling Sentences,» arXiv preprint arXiv:1404.2188, 2014

[4] Wenpeng Yin, Hinrich Schütze, «Multichannel Variable-Size Convolution for Sentence Classification,» arXiv preprint arXiv:1603.04513, 2016

[5] Alexis Conneau, Holger Schwenk, Loïc Barrault, Yann Lecun, «Very Deep Convolutional Networks for Text Classification,» arXiv preprint arXiv:1606.01781, 2017

[6] Sepp Hochreiter, Jurgen Schmidhuber, «Long Short-Term Memory,» 1997

[7] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, «Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translatio,» arXiv preprint arXiv:1406.1078, 2014

[8] https://habr.com/ru/post/487808/

[9] Wenpeng Yin, Katharina Kann, Mo Yu, Hinrich Schutze, «Comparative Study of CNN and RNN for Natural Language Processing,» arXiv preprint arXiv:1702.01923, 2017

[10] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, Francis C.M. Lau, «A C-LSTM Neural Network for Text Classification,» arXiv preprint arXiv:1511.08630, 2015

[11] Sutskever I., «Sequence to Sequence Learning with Neural Networks,» arXiv preprint arXiv:1409.3215, 2014

[12] http://www.davidsbatista.net/blog/2020/01/25/Attention-seq2seq/

[13] Sam Wiseman, Alexander M. Rush., «Sequence-to-Sequence Learning as Beam-Search Optimization,» arXiv preprint arXiv:1606.02960, 2016

[14] https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

[15] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, Francis C.M. Lau, «Neural Machine Translation by Jointly Learning to Align and Translate,» arXiv preprint arXiv:1409.0473, 2016

[16] Jianpeng Cheng, L iDong, Mirella Lapata, «Long Short-Term Memory-Networks for Machine Reading,» arXiv preprint arXiv:1601.06733, 2016

[17] https://www.ibm.com/blogs/watson/2020/11/nlp-vs-nlu-vs-nlg-the-differences-between-three-natural-language-processing-concepts/

[18] Xuezhe Ma and Eduard Hovy, «End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF,» arXiv preprint arXiv:1603.01354, 2016

[19] Subhadip Maji, Arijit Ghosh Chowdhury, Raghav Bali, Vamsi M Bhandaru, «Exclusion and Inclusion - A model agnostic approach to feature importance in DNNs,» arXiv preprint arXiv:2007.16010, 2020

[20] Jonathan H. Clark, Dan Garrette, Iulia Turc, John Wieting, «CANINE: Pre-training an Efficient Tokenization-Free Encoder for Language Representation,» arXiv preprint arXiv:2103.06874, 2021

[21] Paperno, D., Kruszewski, G., Lazaridou, A., Pham, Q. N., Bernardi, R., Pezzelle, S., Baroni, M., Boleda, G., and Fernandez, R., «Thelambada dataset: Word prediction requiring a broad discourse context,» arXiv preprint arXiv:1606.06031, 2016

[22] Hill, F., Bordes, A., Chopra, S., and Weston, J., «The goldilocks principle: Reading children's books with explicit memory representations,» arXiv preprint arXiv:1511.02301, 2015.

[23] Trichelair, P., Emami, A., Cheung, J. C. K., Trischler, A., Suleman, K., and Diaz, F., «On the evaluation of common-sense reasoning in natural language understanding,» arXiv preprint arXiv:1811.01778, 2018.

[24] Reddy, S., Chen, D., and Manning, C. D., «Coqa: A conversational question answering challenge,» arXiv preprint arXiv:1808.07042, 2018

[25] Al-Rfou, R., Choe, D., Constant, N., Guo, M., and Jones, L., «Character-level language modeling with deeper self-attention,» arXiv preprint arXiv:1808.04444, 2018.

[26] Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., «Abstractive text summarization using sequence-to-sequence rnns and beyond,» arXiv preprint arXiv:1602.06023, 2016.

[27] Kwiatkowski, T., Palomaki, J., Rhinehart, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Kelcey, M., Devlin,J., «Natural questions: a benchmark for question answering research.,» 2019.

[28] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R., «Glue: A multi-task benchmark and analysis platform for natural language understanding,» arXiv preprint arXiv:1804.07461, 2018.

[29] McCann, B., Keskar, N. S., Xiong, C., and Socher, R., «The natural language decathlon: Multitask learning as question answering,» arXiv preprint arXiv:1806.08730, 2018

[30] Pranav Rajpurkar, Robin Jia, Percy Liang, «Know What You Don't Know: Unanswerable Questions for SQuAD,» arXiv preprint arXiv:1806.03822, 2018

[31] Danqi Chen, Adam Fisch, Jason Weston and Antoine Bordes, «Reading Wikipedia to Answer Open-Domain Questions,» arXiv preprint arXiv:1704.00051, 2017