МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

ИСПРАВЛЕНИЕ ЗНАКОВ ПУНКТУАЦИИ В ТЕКСТЕ С ПОМОЩЬЮ НЕЙРОННЫХ СЕТЕЙ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Янченко Никиты Александровича

Научный руководитель
к. ф.-м. н., доцент, _____ А. А. Кузнецов

Заведующий кафедрой
к. ф.-м. н., доцент _____ А. С. Иванов

Саратов 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3
1 Теоретическая часть 5
1.1 Long Short-Term Memory (LSTM) 5
1.2 Seq2Seq 6
1.2.1 Seq2Seq модель 6
1.2.2 Механизм внимания (Attention) 6
2 Практическая часть 8
2.1 Выбор данных для обучения 8
2.2 Базовая предобработка данных 8
2.3 Создание датасета и словаря слов 8
2.3.1 Лемматизация 9
2.3.2 Стеммы и флексии 9
2.3.3 Создание батчей 9
2.4 Реализация Bidirectional LSTM 9
2.4.1 Реализация модели 10
2.5 Реализация Bidirectional LSTM Seq2Seq (BiLSTM Seq2Seq) 10
2.5.1 Реализация модели 11
2.6 Подсчет метрик 12
2.7 Эксперименты по обучению 12
ЗАКЛЮЧЕНИЕ 14
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 15

ВВЕДЕНИЕ

Написание писем, сообщений и текстов всегда была неотъемлемой частью жизни людей. Ежедневно каждый человек создает тысячи строк связных предложений и текстов, но не всегда получается соблюдать правила пунктуации русского языка. Некоторые могут быть не критичны для понимания смысла написанного. Однако, возможны случаи, когда неправильно поставленный знак препинания может изменить значение предложения на совершенно противоположное.

Совершение пунктуационных ошибок является одним их самых частых видов нарушений в тексте, а их исправление бывает затруднено даже для грамотного человека. Поэтому исправление знаков пунктуации является важной проблемой на данный момент.

Данная проблема является сложной задачей и в области NLP. Так, она может быть применена в самых разных целях:

— использование в текстовых редакторах для автоматической проверки пунктуации;
— использоваться в сервисах переводов между языками для автоматического проставления знаков препинания;
— восстановление знаков пунктуации в полностью не сегментированных текстах, полученных, например, при распознавании речи [1].

С развитием нейронных сетей в последнее десятилетие удалось достичь огромного прогресса в различных областях, в том числе и в обработке текстовых данных. Так, на данный момент в системе `Google Переводчик` от компании Google уже используются нейронные сети для автоматического исправления знаков препинания.

Поэтому целью данной работы является реализация нейронных моделей и их применение в задачах исправления знаков препинания. Поставленная задача решается в несколько этапов:

— изучение принципов работы рекуррентных сетей, таких как RNN и LSTM как сети, которые наиболее подходят для исправления знаков пунктуации;
— изучение модели Seq2Seq с механизмом внимания;
— реализация архитектур для исправления знаков пунктуации на основе BiLSTM сети и модели Seq2Seq с LSTM сетями и механизмом внимания;

— обучение моделей с различными гиперпараметрами и с разной обработкой тренировочных данных;
— сравнение работы каждой системы, анализ результатов моделей на примерах.

Бакалаврская работа состоит из ведения, трех глав, заключения, списка использованных источников и трёх приложений. Объем работы составляет 43 страницы. Список литературы включает 23 наименования.

В первой главе приведены некоторые теоретические сведения: однонаправленные и двунаправленные рекуррентные сети, главный недостаток стандартной рекуррентной сети, а также основы LSTM сети и архитектуры Seq2Seq с механизмом внимания.

Во второй главе описан выбор данных для обучения и их предварительная обработка, приведены два используемых алгоритма для обработки датасета, представлены реализации нейронных моделей, алгоритмов, необходимые для обучения, и используемых метрик для оценки нейронных сетей, а также результаты обучения.

В третьей главе приведены примеры работ каждой модели, их анализ и предложены дальнейшие варианты разработки и улучшения результатов качества исправления знаков пунктуации с помощью нейронных сетей.

1 Теоретическая часть

1.1 Long Short-Term Memory (LSTM)

LSTM сеть – это разновидность рекуррентной сети, содержащая ячейки памяти специального типа, позволяющие запоминать информацию на короткий или длительный срок. Это происходит, потому что сеть не использует внутри своих рекуррентных компонентов функцию активации. Другими словами, информация не смешивается во времени, а градиент не исчезает при использовании алгоритма обратного распространения ошибки во время обучения. [2].

Ключевой компонент LSTM – это *состояние ячейки (cell state)*, к которой применяется несколько векторных операций. Состояние ячейки проходит через всю ячейку и передает данные через всю цепочку сети. Для управления состоянием ячейки в LSTM сети выделяется структура, называемая «вентили». Существует три вентиля: входной вентиль, вентиль забывания и выходной вентиль.

В общем виде принцип работы LSTM сети можно представить в виде формул [3]:

$$f_t = \sigma(x_t * W^{(f)} + h_{t-1} * U^{(f)} + b^f),$$

$$i_t = \sigma(x_t * W^{(i)} + h_{t-1} * U^{(i)} + b_i)$$

$$\widetilde{C}_t = \tanh(x_t * W^{(C)} + h_{t-1} * U^{(C)} + b_C)$$

$$C_t = f_t \odot C_{t-1} \oplus i_t \odot \widetilde{C}_t$$

$$o_t = \sigma(x_t * W^{(o)} + h_{t-1} * U^{(o)} + b_o)$$

$$h_t = o_t \odot \tanh(_t),$$

где f_t — вентиль забывания; i_t — входной вентиль; $\widetilde{C}_t$ — вектор новых значений

состояния ячейки; C_t — новое состояние ячейки; o_t — выходной вентиль; h_t — выходные данные ячейки;

1.2 Seq2Seq

1.2.1 Seq2Seq модель

Архитектура Seq2Seq состоит из двух отдельных сетей: *кодировщик (encoder)*, который обрабатывает входные данные и *декодировщик (decoder)*, который генерирует выходные данные.

Кодировщик, получая входную последовательность слов, преобразует их во входные вектора. Для этого используется *матрица представлений (embedding)*. Полученные строки матрицы обрабатываются энкодером и скрытые состояния кодировщика передаются в декодировщик.

Декодировщик получает выходные данные из кодировщика и использует их как свои начальные состояния для обработки, пытаясь восстановить целевую последовательность [4].

Главной проблемой Seq2Seq моделей является необходимость сжать слова произвольной длины в вектор представлений фиксированного размера. Это трудность становится существенной при больших входных последовательностях. Для решения такой проблемы в моделях принято использовать *механизмы внимания*.

1.2.2 Механизм внимания (Attention)

Механизм внимания — это метод в обучении, который заключается в выделении части входных данных для более тщательной обработки. Механизм внимания представляет собой отдельную небольшую нейронную сеть, где на каждом временном шаге t декодера высчитывается *вес внимания* w_{tj} для скрытого состояния h_j кодировщика, который означает вероятность того, что внимание должно быть сфокусировано именно на этом состоянии:

$$e_{tj} = a(h_j, s_{t-1}), \forall j \in [1, T], \tag{1}$$

$$w_{tj} = \frac{\exp(e_{tj})}{\sum\limits_{K=1}^{T} \exp(e_{tk})}, \tag{2}$$

, где e_{tj} — мера близости; h_j — скрытое состояние кодировщика; s_{t-1} — предыдущее скрытое состояние декодировщика, a — параметр, который параметризован как нейронная сеть с прямой связью (FeedForward Network), работающая для всех j на шаге времени декодирования t. На следующем шаге строится вектор контекста $\hat{h_t}$, который является линейной комбинацией скрытых состояний кодировщика и веса внимания: [5]:

$$a_{tj} = softmax(w_{tj}), \tag{3}$$

$$\hat{h_t} = \sum_{j=1}^{T} a_{tj} h_j, \tag{4}$$

На последнем шаге вектор контекста $\hat{h_t}$ подается в декодер, в котором он конкатенируется с предыдущими скрытыми состояниями декодировщика s_{t-1} и выводится состояние s_t.

Данный подход вычисления называется методом *мягкого внимания*. Самым популярным механизмом мягкого внимания является *Bahdanau Attention*. Благодаря механизму внимания, существенно улучшается работа модели на больших данных.

2 Практическая часть

В рамках данной работы было реализовано построение и обучение нейронных сетей для исправления знаков пунктуации на основе архитектур BiLSTM и Seq2Seq с BiLSTM сетью в качестве энкодера и LSTM сети как декодер.

Обучение происходило для следующих знаков препинания: *точка*, *запятая*, *знак вопроса*, *восклицательный знак*, *двоеточие*, *тире* и *пробел*, как показатель, что никакого знака пунктуации после слова нет.

Для обучения использовалась библиотека машинного обучения `Pytorch` [6].

2.1 Выбор данных для обучения

Для создания обучающегося датасета был использован сборник литературных произведений различных направлений и жанров. В общей сложности использовались тексты из 152 книг.

2.2 Базовая предобработка данных

Предварительно все книги обрабатывались следующим образом:

- перевод в нижний регистр;
- буква «ё» заменена на «е»;
- перевод в кодировку UTF—8;
- удаление символов, которые не относятся к буквам русского алфавита и знакам препинания;
- все данные в тексте, которые представлены в виде последовательности цифр, такие как целые или дробные числа, а также отображение времени, заменяются на специальный токен;
- разделение слов и знаков препинания пробелами;
- замена всех повторяющихся знаков препинания одинарными.

После обработки, тексты разбивались на отдельные абзацы. Если количество слов и знаков препинания в абзаце было меньше 100, то абзац брался целиком. Иначе, абзац разбивался на отдельные предложения.

2.3 Создание датасета и словаря слов

Для уменьшения размера словаря слов датасета можно применить несколько алгоритмов обработки, такие как использование лемматизации или разделение слов на стеммы и флексии. Для сравнения влияния алгоритмов на качество

нейронных сетей, было решено использовать оба алгоритма в обучении.

2.3.1 Лемматизация

Лемматизация - это приведение слова в предложениях к начальной форме. Такая обработка позволяет многократно сократить размер словаря и избавиться от грамматических форм слов, которые будут влиять на качество обучения, но при этом оставить контекст слова.

Минусом лемматизации является то, что применение правил приведение к начальной форме выполняется и для различных частей речи. Так, например, глаголы, причастия и деепричастия приводятся к единой начальной форме - глагол в несовершенном виде. Лемматизация датасета проводилась с помощью библиотеки для морфологического анализа текста `MyStem` [7].

2.3.2 Стеммы и флексии

Разбиение на стеммы и флексии - это способ обработки, при котором каждое слово разделяется на две части: основа, называемая стеммой, и изменяемая часть - флексия. Таким образом, каждая часть заносится в словарь и рассматривается как отдельное слово. Такая обработка позволяет существенно сократить размер словаря и сохранить различные формы слов, в том числе и в разных частях речи. Обработка датасета с разделением на стеммы и флексии проводилась с помощью стеммера `Snowball`, который находится в библитеке `NLTK` [8].

2.3.3 Создание батчей

Для создания батчей был создан экземпляр класса `DataLoader` из библиотеки машинного обучения Pytorch, который состоит из экземпляра класса `LangDataset`, `RandomBatchSampler` и `CollateFn`.

В классе `LangDataset` хранится датасет, словарь токенов и словарь знаков пунктуации, где каждый токен и знак закодирован числовым индексом. Класс `RandomBatchSampler` позволяет генерировать мини-батчи со случайными предложениями, а класс `CollateFn` производит обработку этого мини-батча, которые будут подаваться в нейронные сети.

2.4 Реализация Bidirectional LSTM

Первой реализованной нейронной сетью была модель с двунаправленной LSTM сетью, архитектура которой показана на рисунке 1.

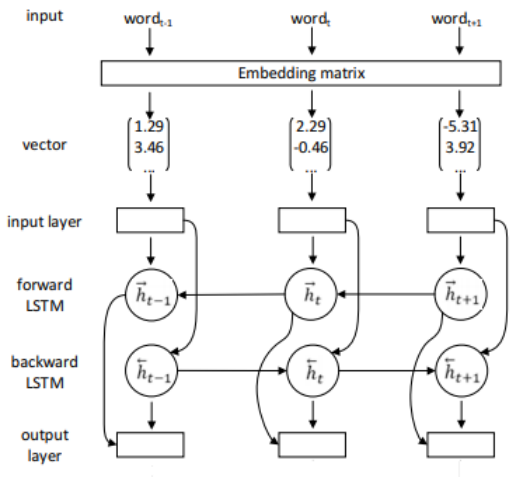

Рисунок 1 – Модель архитектуры BiLSTM

Двунаправленная LSTM сеть позволяет считывать текстовую последовательность в прямом и обратном направлении. Это необходимо для лучшего учета контекста, так как каждое слово зависит как от предыдущих, так и от последующих слов в предложении.

2.4.1 Реализация модели

Модель представлена классом `BiLSTM`. На вход подаются данные, которые преобразуются в векторное представление и упаковываются для лучшей и эффективной работы с данными различной длины. Вычисляются выходные значения модели и подаются в полносвязный слой, который переводит их в размерность выходного словаря, т. е количество знаков пунктуации.

В качестве оптимизатора выступает `Adam` [9], функцией потери является `CrossEntropyLoss`.

2.5 Реализация Bidirectional LSTM Seq2Seq (BiLSTM Seq2Seq)

Второй реализованной нейронной сетью была модель Seq2Seq с механизмом внимания. В качестве энкодера использовалась BiLSTM сеть. Выбор в пользу использования двунаправленной сети был сделан, чтобы также улучшить связь и контекст между словами. В качестве декодера использовалась LSTM сеть с механизмом внимания Bahdanau Attention.

Архитектура данной модели представлена на рисунке 2

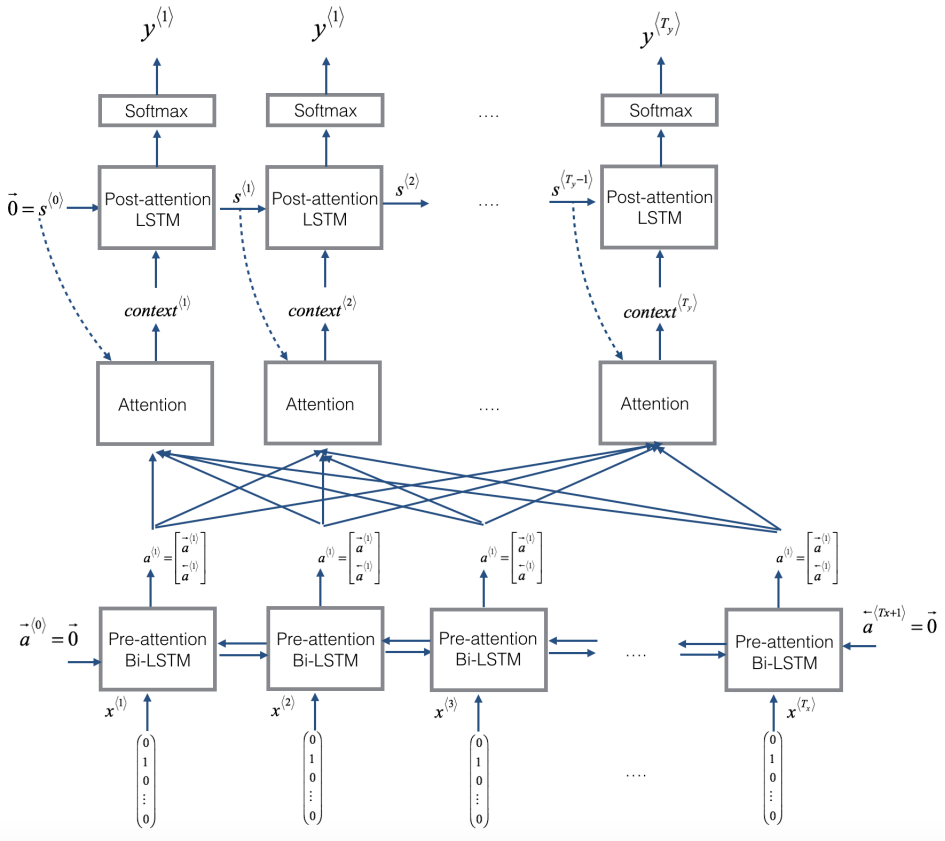

Рисунок 2 – Модель архитектуры BiLSTM Seq2Seq

2.5.1 Реализация модели

Модель представлена классами `EncoderLSTM`, `BahdanauAttention`, `DecoderLSTM_with_BahdanauAttention`, `Generator` и `Seq2Seq`.

Класс `EncoderLSTM` реализовывает энкодер в модели Seq2Seq, в нем обучается сеть на основе входных последовательностей и передает свой выход и скрытые состояния прямо декодеру. Так как в энкодере используется двунаправленная LSTM сеть, а в декодере однонаправленная, то скрытые состояния сначала конкатенируются, чтобы уменьшить размерность в два раза.

Класс `BahdanauAttention` реализовывает механизм внимания, который высчитывается на основе скрытых состояний из энкодера, и строит вектор контекста, что уже подается в декодер для конкатенации с предыдущими скрытыми состояниями декодера.

Класс `DecoderLSTM_with_BahdanauAttention` реализовывает декодер с механизмом внимания модели Seq2Seq. Он получает выходные последовательности из энкодера и контекстный вектор из механизма внимания и возвращает выходную последовательность размерности словаря знаков препинания

Класс `Seq2Seq` реализовывает архитектуру Seq2Seq, инициализирует эн-

кодер и декодер. Экземпляр этого класса, которому подаются входные и выходные последовательности из генератора батча, используется во время обучения.

Класс `Generator` представляет собой небольшую структуру, которая пропускает выходные последовательности из декодера через полносвязный слой и функции `log_softmax`, чтобы выдать токен с максимальным весом. Используется в реализации жадного поиска для декодирования значений декодера.

В качестве оптимизатора градиентного спуска выступает `Adam` [9], функцией потери является `CrossEntropyLoss`.

2.6 Подсчет метрик

Для оценивания качества работы моделей использовались метрики `Precision`, `Recall` и `F-Score` каждого из знаков пунктуации в частности, и `Accuracy` как общая для всех. Для их подсчета использовался метод `precision_recall_fscore_support` из библиотеки `sklearn` [10].

2.7 Эксперименты по обучению

Для сравнения качества работы архитектур были обучены:

— 3 модели с обработкой датасета лемматизацией: 2 модели BiLSTM с разными гиперпараметрами и одна BiLSTM Seq2Seq.

— 2 модели с обработкой датасета стеммами и флексиями: BiLSTM и BiLSTM Seq2Seq.

Гиперпараметры моделей представлены в таблице 1.

Таблица 1 – Гиперпараметры реализуемых моделей

Гипер-параметры	BiLSTM №1 (лемматизация)	BiLSTM №2 (лемматизация)	BiLSTM №3 (стеммы и флексии)	BiLSTM Seq2Seq №1 (лемматизация)	BiLSTM Seq2Seq №2 (стеммы и флексии)
Batch Size	128	128	128	128	32
Num Layers	2	2	2	2	2
Embedding Size	128	256	256	256	256
Hidden Size	256	512	512	512	512
Dropout	0.2	0.2	0.2	0.2	0.2
Learning Rate	0.0001	0.0001	0.0001	0.0001	0.0001

По итогам, лучшей моделью по большинству знаков пунктуации и общему значению метрик оказалась архитектура BiLSTM Seq2Seq. Что касается сравнения моделей по способу обработки датасета, то можно сказать, что в среднем лемматизация и стеммы показывают себя с одинаковой точностью, но при обработке с помощью стемм и флексий модели обучаются дольше.

Так же интересны результаты работы моделей для `восклицательного` и `вопросительного знаков`. По итогам анализа примеров можно сказать, что нейронной сети тяжело различить, какой контекст может нести одно предложение: вопросительный или восклицательный. Особенно в русском языке, где, в отличии от английского, нет особых правил построения предложений для вопросов. Однако стоит заметить, что и человеку будет затруднительно определить постановку знака, основываясь только на контексте предложения, потому что существует мало способов передачи повышенной интонации с помощью слов на письме.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы были выполнены поставленные задачи:

- изучены принципы работы RNN и LSTM сетей;
- изучена модель Seq2Seq с механизмом внимания;
- реализованы архитектуры для исправления знаков пунктуации на основе BiLSTM сети и модели Seq2Seq с LSTM сетями и механизмом внимания;
- обучены модели с различными гиперпараметрами и с разной обработкой тренировочных данных;
- произведено сравнение работы каждой системы, проанализированы результаты моделей на примерах;
- предложены возможные пути улучшения качества выполнения задачи исправления знаков пунктуации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Tilk, O.* LSTM for punctuation restoration in speech transcripts // Interspeech 2015. — Dresden, Germany: 2015. — Pp. 683–687.

2 *Гудфеллоу, Я.* Глубокое обучение = Deep Learning / Я. Гудфеллоу, И. Бенджио, А. Курвилль. — М.: ДМК Пресс, 2017.

3 *Greff, K.* LSTM: A search space odyssey / K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, J. Schmidhuber // *CoRR*. — 2015. — Vol. abs/1503.04069.

4 *Sutskever, I.* Sequence to sequence learning with neural networks / I. Sutskever, O. Vinyals, Q. V. Le. — 2014.

5 *Bahdanau, D.* Neural machine translation by jointly learning to align and translate / D. Bahdanau, K. Cho, Y. Bengio // *arXiv e-prints*. — September 2014. — Vol. abs/1409.0473.

6 Pytorch: An imperative style, high-performance deep learning library / A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen et al. — 2019. — Pp. 8024–8035.

7 Яндекс технология MyStem. [Электронный ресурс]. — URL: https://tech.yandex.ru/mystem/ (Дата обращения 19.05.2020). Загл. с экр. Яз. рус.

8 *Loper, E.* Nltk: The natural language toolkit // In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics. — 2002.

9 *Kingma, D. P.* Adam: A method for stochastic optimization / D. P. Kingma, J. Ba // *CoRR*. — 2015. — Vol. abs/1412.6980.

10 Scikit-learn: Machine learning in Python / F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel et al. // *Journal of Machine Learning Research*. — 2011. — Vol. 12. — Pp. 2825–2830.