The Algorithm of Automatic Accentuation with Respect to the Speaking Norm of a Given Author

Anna Mosolova

Institute of Computational Technologies, SB RAS
a.mosolova@g.nsu.ru

Abstract. The task of automatic accentuation is important to many fields of natural language processing including speech generation and poetry generation. In this paper we introduce a deep learning based algorithm of automatic accentuation. We describe the creation of a dataset from a collection of works by A. S. Pushkin, a grammar dictionary by A. A. Zaliznyak and a concordance of Pushkin's poems, the training of a recurrent neural network on this dataset and the evaluation of the resulting algorithm. The described algorithm outperforms the CRF-based baseline by 10%.

Алгоритм автоматической акцентуации с учетом орфоэпической нормы автора*

А. В. Мосолова

Институт вычислительных технологий СО РАН
a.mosolova@g.nsu.ru

Аннотация Задача автоматической расстановки ударений важна для многих разделов обработки естественного языка, включая синтез речи и генерацию стихотворного текста. В данной статье мы предлагаем основанный на глубоком обучении алгоритм расстановки ударений. В статье описывается составление набора данных с помощью собрания сочинений А. С. Пушкина, грамматического словаря А. А. Зализняка и конкорданса к стихам А. С. Пушкина, обучение на этом наборе данных рекуррентной нейронной сети и исследование качества работы полученного алгоритма. Описываемый алгоритм превосходит базовый метод, основанный на алгоритме условных случайных полей, на десять процентов.

Keywords: Акцентуация · Рекуррентная нейронная сеть

1 Введение

Во многих областях анализа текста для его корректного рассмотрения требуется наличие ударений у слов. Особенно это важно при анализе стихов. Однако стандартные алгоритмы акцентуации проставляют ударения на основе современных норм русского языка, в то время как ритмический рисунок и рифма в стихотворениях многих авторов работает только при той орфоэпической норме, которая была принята в тот временной период развития русского языка, когда они создавали свои произведения. Таким образом, мы видим своем задачей разработку алгоритмов для автоматической расстановки ударений, который проводит акцентуацию с учетом норм, которые были использованы автором при написании своего произведения. В данной работе будут представлены два алгоритма, решающие эту задачу, имитируя нормы, которые использовал А. С. Пушкин. В разделе о данных опишем подготовку наборов данных, основанных на произведениях А. С. Пушкина, для обучения алгоритмов, далее рассмотрим устройство алгоритма с использованием условных случайных полей и алгоритма с рекуррентной нейронной сетью. В конце представлены результаты оценки качества работы этих алгоритмов на тестовом множестве и заключение о работе.

* Работа осуществлена в рамках гранта РНФ № 19-18-00466 «Разработка и реализация информационной системы многоуровнего исследования стихотворных текстов»

2 Данные

Исходный набор данных был обработан двумя способами для тестирования каждого из предложенных в статье алгоритмов. Для первого алгоритма, который использовался в качестве базового метода, был создан специальный корпус, основанный на первых четырёх томах собрания сочинений А. С. Пушкина в десяти томах [1]. Во всех стихотворениях были проставлены ударения с помощью акцентора, основанного на грамматическом словаре русского языка А. А. Зализняка (ударения проставлялись в соответствии с типом акцентной парадигмы из словаря), который использовал не только саму словоформу, но и ее морфологические характеристики для корректного проставления ударения в одинаковых словоформах некоторой лексемы [4]. Также был использован «Конкорданс к стихам А. С. Пушкина» Дж. Томаса Шоу [2,3], из которого брались ударения к словам в строках, где встречались омографы (рис. 1). Те предложения, в которых ударение не могло было быть проставлено ввиду отсутствия какой-то из словоформ и в словаре А. А. Зализняка, и в Конкордансе, удалялись из стихотворения. Обработанные таким образом стихотворения затем преобразовывались в пары последовательностей, состоящие из входной последовательности букв и желаемой выходной последовательности нулей и единиц, причём единица для элемента желаемой выходной последовательности означает то, что соответствующий этому элементу символ входной последовательности должен находиться под ударением, а ноль - что нет (рис. 2).

Первая из последовательностей поступала на вход алгоритму, который делал ее предобработку перед использованием в качестве обучающего множества для нейросети. Каждый символ получал свой набор признаков: является ли он заглавной/строчной буквой, буквой/цифрой/знаком препинания, предыдущая перед текущим символом буква, следующая за текущим символом буква. Для другого варианта алгоритма расстановки ударений, осно-

Рис. 1.

ванного на проставлении ударения не для целого текста, а для одного слова, использовался другой способ представления данных. Здесь на вход подается

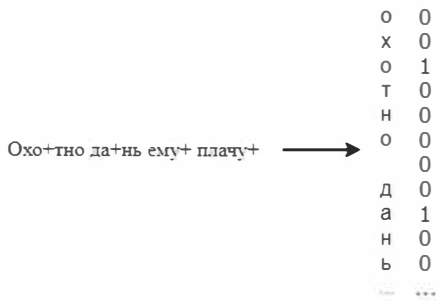

Рис. 2.

цепочка символов слова, а также его морфологическая характеристика, что позволяет автоматически разрешать неодзнозначность в омографах.

При обучении второго алгоритма использовался метод переноса обучения (transfer learning), поэтому для первичного обучения также был использован корпус прозаических текстов, в которых были расставлены ударения. Количество уникальных слов в обучающей выборке составило более 800 тысяч.

Дообучение проводилось с помощью второго массива данных - набора слов из [2,3].

Морфологическая информация о каждом слове генерировалась автоматически на основе словарного метода с использованием библиотеки pymorphy2 [6]. Омография на этапе подготовки корпуса разрешалась вручную.

3 Алгоритм

3.1 Базовый метод: условные случайные поля

Обработанные описанным в разделе 2 способом последовательности поступали на вход алгоритма, использующего метод условных случайных полей (Conditional random fields, CRF), который является разновидностью метода Марковских случайных полей [5]. Принцип работы CRF схож с логистической регрессией, этим объясняется быстрое обучение моделей такого рода, однако она предназначена для предсказания последовательностей и способна использовать, в отличие от логистической регрессии, контекст в качестве признаков. Метод условных случайных полей часто применяется для задач определения частей речи, распознавания именованных сущностей, определения элементов последовательностей изображений, поэтому должен хорошо себя показывать и в задаче предсказания ударения.

Для обучения алгоритма акцентуации мы использовали имплементацию CRF в библиотеке sklearn-crfsuite [8] из библиотеки sklearn [7] для подбора оптимального алгоритма обучения, а также коэффициента регуляризации.

3.2 Рекуррентная нейронная сеть

Второй алгоритм использует для решения задачи классификации на ударные и безударные позиции рекуррентные нейронные сети типа GRU (Gated recurrent unit).

Если смотреть на рекуррентную нейронную сеть с архитектурной точки зрения, то на вход ей подается некоторая последовательность $x_i, ...; x_j$, каждая из которых представляет собой вектор размерности d длины n. На выходе RNN выдает один вектор размерности d. Таким образом, ее можно представить в виде функции $y_n = RNN(x_{1:n})$, где $y_i = RNN(x_{1:i})$. Выходной вектор y_n используется для предсказания следующей части последовательности, из этого становится понятно, что в RNN используется некоторая рекурсия,так как для предсказания вектора состояния s_i она использует и вектор s_{i-1}, который является выходом предыдущего состояния, и вектор x_i, который является непосредственным входом для этого состояния. Вектор состояния s_i затем отображается в выходной вектор y_i, используя функцию $O()$.

Таким образом:

$$RNN(x_{1:n}; s_0) = y_{1:n},$$
$$\mathrm{y}_i = O(s_i),$$
$$\mathrm{s}_i = R(s_{i-1}, x_i),$$

причем функции R и O едины для всей RNN. В конечном выходе RNN, таким образом, зафиксирована часть информации из всех входных векторов и из выходов всех состояний, что оказывается очень полезным в большом числе прикладных задач.

Что касается обучения RNN, то для этого используется процедура, именуемая backpropagation through time, описанная в [9] и реализующая подбор коэффициентов таких, которые бы минимизировали кросс-энтропийную функцию ошибки. Для этого рекуррентная сеть разворачивается в сеть прямого распространения. Затем в три этапа считается градиент. На первом этапе вычисляются состояния слоев, на втором ошибки слоев, а на третьем изменения весов для следующей итерации.

Главной проблемой простой рекуррентной сети является тот факт, что через определенное число шагов она “забывает” то, что происходило на первом шаге, и это, например, является затруднением при обработке текстов на естественном языке, так как не позволяет учитывать анафору, а также предыдущий контекст. Для решения этой проблемы были созданы архитектуры, основанные на так называемых вентилях. Идея состоит в том, что в обычной RNN мы не можем контролировать запоминание и забывание информации, но если предположить, что у нас есть бинарный вектор $g \in \{0, 1\}^n$, то потерю информации можно сдерживать. Такой вектор мог бы быть вентилем для контроля к векторам размерности n, используя произведение Адамара. Пусть память (состояние) $s \in R^d$, входной вектор $x \in R^d$

и вентиль $g \in \{0,1\}^d$, тогда s' вычисляется через произведение Адамара g и x и его суммированием с тем же произведением $(1-g)$ на s.

Таким образом мы можем контролировать запоминание отдельных элементов, однако для встраивания подобного вектора необходимо, чтобы мы могли дифференцировать функцию, которая выдавала бы нам 1 или 0. Для этих целей используется вектор $g' \in R^n$, который затем передается в сигмоидную функцию для получения нужных 1 или 0.

Первой архитектурой, использующей преобразование такого рода, была рекуррентная нейронная сеть типа LSTM [10], однако требуется долгое время для приобретения этой сетью обобщающей способности, поэтому нами была использована рекуррентная нейронная сеть типа GRU.

Рекуррентный слой при обработке нового элемента последовательности получает на вход не только входной вектор признаков, но и вектор состояния слоя в предыдущий момент времени, поэтому было решено передавать информацию о морфологической характеристике слова вместо вектора состояния слоя в предыдущий момент времени, так как простая конкатенация слова и его морфологической характеристики значительно увеличивает размерность вектора признаков и, как следствие, экспоненциально растёт число обучаемых параметров нейронной сети, что может привести к её переобучению, т.е. потере обобщающей способности. Обычно в первый момент времени (при обработке первого элемента последовательности) вектор состояния считается нулевым. Мы же будем инициализировать вектор состояния для первого элемента последовательности не нулями, а признаковым описанием морфологической характеристики слова. Сама морфологическая характеристика слова прндварительно получается с помощью основанного на нейросети морфоанализатора RNNMorph [6]. Для обучения данной нейронной сети был применён метод переноса обучения (transfer learning). Сначала нейросеть была обучена расставлять ударения в словах, полученных из обычных прозаических текстов, по правилам современного русского языка. Затем сеть была дообучена на словах из [2,3].

4 Результаты

Тестирование алгоритма проводилось на 25% от полученных наборов данных.

В результате экспериментов было показано, что вариант алгоритма на основе рекуррентных нейронных сетей типа GRU, инициализируемых состоянием на основе векторного представления морфохарактеристик слова, показывает 5,6% ошибок в расстановке ударений. Базовый алгоритм, основанный на условных случайных полях с использованием признаков символов, дает 15% ошибок на тестовом множестве (табл. 1).

Алгоритм, основанный на рекуррентных нейронных сетях, показывает лучшее качество по сравнению с условными случайными полями, в связи со своей способностью улавливать более тонкие закономерности в подаваемых ему на вход последовательностях. Однако, если необходимо получить

Таблица 1. Результаты

Алгоритм	% ошибок
Условные случайные поля (базовый метод)	15
Рекуррентная нейронная сеть	5,6

алгоритм, который способен быстро обучаться, выбор будет отдан (хоть и с некоторой потерей качества) в пользу CRF.

5 Заключение

В данной статье мы описали построение корпуса для обучения алгоритма автоматической расстановки ударения, базовый алгоритм, основанный на условных случайных полях, и алгоритм, использующий рекуррентную нейронную сеть и морфологические характеристики слова в качестве скрытого состояния. Эксперименты показали, что последний алгоритм выполняет задачу акцентуации на 10% лучше, чем базовый метод.

Благодарности. Эта работа выполнена под руководством Ивана Юрьевича Бондаренко в рамках работы над грантом РНФ № 19-18-00466 «Разработка и реализация информационной системы многоуровнего исследования стихотворных текстов».

Список литературы

1. Пушкин, А. С., Данилова, А. М.: Собрание сочинений в десяти томах. Вагриус, Москва (2005)
2. Шоу, Дж. Т.: Конкорданс к стихам А. С. Пушкина. Т. 1. Языки русской культуры, Москва (2000)
3. Шоу, Дж. Т.: Конкорданс к стихам А. С. Пушкина. Т. 2. Языки русской культуры, Москва (2000)
4. Yakovenko, O., et al.: Algorithms for automatic accentuation and transcription of russian texts in speech recognition systems. In: In proceedings of the International Conference on Speech and Computer. pp. 768-777. Springer, Cham (2018)
5. Sutton, Ch., McCallum, A. An introduction to conditional random fields. In: Foundations and Trends® in Machine Learning. pp. 267-373. Now Publishers Inc., Hanover (2012)
6. Anastasyev D. G., Gusev I. O., Indenbom E. M.: Improving Part-of-speech Tagging Via Multi-task Learning and Character-level Word Representations. In: Proceedings of the International Conference "Dialogue 2018". pp. 14-27. Moscow (2018)
7. Pedregosa F. et al.: Scikit-learn: Machine learning in Python. In: The Journal of machine Learning research. pp. 2825-2830. (2011)
8. Okazaki N.: Crfsuite: a fast implementation of conditional random fields (crfs). (2007)

9. Werbos P. J.: Backpropagation through time: what it does and how to do it. In: Proceedings of the IEEE. pp. 1550-1560. (1990)
10. Hochreiter S., Schmidhuber J.: Long short-term memory. In: Neural computation. pp. 1735-1780. (1997)