Разработка и реализация методов генерации правил для автоматической проверки правописания

П.Я. Бахвалов

Университет ИТМО

pavelbakhv@gmail.com

Аннотация

Задача автоматической проверки правописания является актуальной: Количество написанного текста увеличивается с каждым годом, также как и количество людей, начинающих изучать новые языки, и всем нам, как людям, свойственно делать ошибки.

Существует два основных подхода к решению данной задачи: подход, основанный на машинном обучении, и подход, основанный на правилах. Первый имеет более высокое качество и не требует привлечение лингвистов, но с другой стороны второй позволяет объяснить пользователю причину ошибки и требует значительно меньше вычислительных ресурсов. Эти подходы можно комбинировать, объединяя их преимущества, и получать выигрыш в качестве.

В настоящей работе представлен способ автоматического получения правил из аннотированного набора данных, которыми могут быть расширены системы автоматической проверки правописания после добавления описания.

За основу был взят подход Transformation-Based Learning (TBL), который был доработан для использования на данных с большим количеством признаков. В результате были сгенерированы 1238 правил для 36 категорий ошибок. После этого, существующая система по проверке правописания LanguageTool была расширена полученными правилами и показала улучшение качества работы.

Ключевые слова: проверка правописания, автоматическое исправление ошибок, генерация правил, детектирование ошибок

Библиографическая ссылка: Бахвалов П.Я. Разработка и реализация методов генерации правил для автоматической проверки правописания // Компьютерная лингвистика и вычислительные онтологии. Выпуск 4 (Труды XXIII Международной объединенной научной конференции «Интернет и современное общество», IMS-2020, Санкт-Петербург, 17 – 20 июня 2020 г. Сборник научных статей). – СПб.: Университет ИТМО, 2020. С. 83-97. DOI: 10.17586/0000-0000-2020-4-83-97

Введение

Автоматическая проверка правописания – это задача автоматического обнаружения и исправления грамматических, стилистических и орфографических ошибок в тексте. Актуальность этой задачи определяется тем, что людям свойственно делать ошибки правописания. Далеко не все хорошо знакомы с правилами грамматики языка или только приступают к их изучению. Помимо этого, существуют болезни, такие как дисграфия, при которых человеку особенно тяжело правильно писать. Задача осложняется за счет таких факторов как неформализованная грамматика, свободный порядок слов в предложении, зависимость слов от контекста, постоянные изменения в языке, диалекты, сленг, жаргон, омонимы и др.

Подходы к задаче автоматической проверки правописания можно разделить на три типа: основанные на правилах, основанные на методах машинного обучения и гибридные. Подход, основанный на правилах, считается устаревшим, т.к. правила обычно пишут квалифицированные лингвисты, имеет низкую полноту, но более высокую точность. С другой стороны, подходы на основе машинного обучения показывают высокое покрытие ошибок и сейчас являются state-of-the-art. Гибридные же подходы обычно выигрывают, т.к. всегда можно взять за основу подход на машинном обучении и добавить в него правила, для которых взятый подход плохо работает, тем самым увеличивая точность и полноту.

Несмотря на значительный прогресс в методах решения задачи проверки правописания, основанных на машинном обучении, у подхода, основанного на правилах, остается ряд существенных преимуществ:

— интерпретируемость правил;
— возможность редактирования набора правил;
— низкая потребность в вычислительных ресурсах.

Квалифицированный специалист всегда может посмотреть на каждое конкретное правило и определить, соответствует ли оно правилу в естественном языке. Пользователь системы также сможет получить информацию о правиле, которое он нарушил.

В системе, основанной на правилах, новые правила можно просто добавить в общий список. То же самое относится к ошибкам в правилах: лингвист сможет исправить правило, добавить исключение или выключить его в такой системе, в то время как в системе, основанной на машинном обучении, ошибка будет возникать снова и необходимо полностью переобучать модель.

Системы, основанные на машинном обучении, гораздо более требовательны к ресурсам, поэтому их невозможно использовать локально на устройствах пользователя. К сожалению, не все пользователи имеют доступ к безлимитному и бесперебойному интернету для взаимодействия с такими системами на выделенных серверах; особые случаи связаны с работой с конфиденциальной информацией, где недопустима отправка данных по сети, в особенности на сторонний сервер. Модель, которая заняла первое место на последнем соревновании по автоматическому исправлению ошибок (BEA-2019) [1], занимает 3.7Гб в сжатом виде, в то время как системы, основанные на правилах, занимают всего десятки мегабайт.

Основной сложностью систем, основанной на правилах, является то, что правила пишутся вручную квалифицированными лингвистами. Когда количество правил достигает тысяч, становится проблематично отслеживать полноту правил и проверять, покрывает ли каждое правило все случаи, в том числе исключения. Решением этих проблем может быть автоматическая генерация правил. Существует ряд работ в этом направлении. Авторы [2] предлагают подход, основанный на имитации иммунной системы человека, когда ошибки рассматриваются как патогены, а правила как лимфоциты. Существуют подходы, основанные на расстоянии редактирования на уровне слов [3]. Однако ни один из подходов не работает хорошо на всех типах ошибок. Большинство описанных в литературе реализаций инструментов недоступно для публичного использования. К тому же сложно сравнивать результаты разных методов, т.к. в каждой работе используется свой набор данных.

На практике системы проверки правописания на основе правил активно используются и развиваются в настоящее время. Одной из таких систем является LanguageTool, для которого с 2003 года команда лингвистов пишет правила на основе присылаемых им ошибок [4]. Данная система является бесплатной и имеет открытый исходный код, а также подходит для автономного использования на локальном компьютере, поэтому набор правил данной системы был взят за основу настоящей работы и сгенерированные правила в итоге будут адаптированы под неё.

В настоящей работе был выбран английский язык, т.к. он является основным языком научных коммуникаций и технической документации.

Авторы обзора систем автоматической проверки грамматики предлагают следующую классификацию типов ошибок [5]: грамматика, пунктуация, структура предложения, орфография, семантика.

В данной работе будут рассматриваться только ошибки грамматики, пунктуации и ошибки в структуре предложения. Исправление орфографических ошибок хорошо разработанная область, точность исправления таких ошибок уже достигает 91% [6]. Категория семантики, с другой стороны, слишком сложна, и предложения с такими ошибками имеют правильную грамматическую структуру. Для определения такой ошибки часто требуется учитывать контекст нескольких предложений, а иногда и целого текста. Как исключение, есть редкая подкатегория ошибок, когда люди используют похожее по смыслу слово, но которое недопустимо в контексте данного словосочетания. Например: «This is a durable (strong) drink». Но такие ошибки исправляются по словарю подобных словосочетаний и не входят в тематику настоящей работы. Поэтому задача генерации правил для ошибок из категории семантики не имеет смысла.

1. Обзор существующих методов

Исследования по автоматическому исправлению правописания ведутся с 60-х годов 20 века. Уже тогда были попытки решения задачи с использованием обычного словаря, n-граммной модели, и языковых моделей [7]. Новый импульс исследованиям придали открытые соревнования, которые проводятся с 2011 года. Первым в своем роде соревнованием является HOO-2011, который предоставил первый общий набор данных для работы с ошибками правописания в открытый доступ, а также общую систему оценивания работ [8]. Потом был CoNLL-2014 [9] и последний из таких контестов BEA-2019, набор данных и система проверки которого были использованы в настоящей работе для проверки качества полученных результатов [1].

В 2011 году подход на основе байесовских классификаторов для исправления ошибок в артиклях, предлогах и некорректном выборе слов продемонстрировал превосходство над подходами, основанными на правилах [10]. В 2014 году лучшие результаты показали подходы, основанные на статистическом машинном переводе [11]. Далее стали появляться подходы, основанные на использовании языковых моделей [12; 13]. Позже для решения задачи исправления ошибок в тексте стали использоваться более сложные модели машинного обучения, например, трансформеры [14; 15]. Лучшим подходом на текущий момент на данных BEA-2019 является система, представляющая собой ансамбль моделей других участников соревнования [16].

2. Подготовка данных для экспериментов

2.1. Исходные данные

В качестве набора данных были использованы данные, предоставленные соревнованием BEA-2019, оно состоит из двух частей. Первая часть – данные с онлайн-платформы Write & Improve [17], которая предлагает помощь при написании текстов людям, для которых английский язык не является родным. Вторая часть набора данных – The LOCNESS corpus [18] – сборник сочинений студентов-носителей английского языка, размеченный сотрудниками Write & Improve.


```
{
 "text": "I think that the public transport will always be in the future.",
 "edits": [{"from":13,"to":16,"fix":","}]
},
```


Рис. 1. Пример из набора размеченных данных

Общий объем данных – примерно 40000 предложений с ошибками и их исправлениями, сделанными одновременно и носителями, и не носителями языка [1]. Пример из набора данных представлен на рисунке 1.

2.2. Разделение по категориям ошибок

На начальном этапе необходимо было разделить данные на группы по типам ошибок. Для данной задачи был использован инструмент для аннотации грамматических ошибок ERRANT [19]. В результате все ошибки были поделены по типам и по категориям (см. табл. 1 и 2).

Таблица 1. Категории грамматических ошибок

Категория	Описание
Части речи	Все слова до и после исправления имеют одинаковую часть речи и не соответствуют критериям для более конкретного типа (consuming ➔ to eat). Сюда же входят ошибки пунктуации.
Сокращения	Хотя бы одно слово до или после исправления является сокращением ('d,' ll, 'm, n't, re,' s или 've), имеется не более одного слова до или после исправления, а также все слова имеют одинаковую часть речи.
Морфология	В ошибке до и после исправления находится ровно одно слово, слова имеют одну и ту же лемму, но больше ничего общего (quick (ADJ) ➔ quickly (ADV)).
Орфография	Слова до и после исправления ошибки после перевода в нижний регистр и удаления всех пробелов становятся одинаковыми (firstly ➔ Firstly).
Опечатки	Слово не находится в словаре Hunspell и исправление содержит 50% букв исходного слова в том же порядке (greatful ➔ grateful).
Неверный порядок	Исправление является перестановкой слов в ошибке (house white ➔ white house).
Форма прилагательного	В ошибке и исправлении ровно одно слово, и оба слова имеют одинаковую лемму, а также являются прилагательными. Либо в ошибке, либо в исправлении два слова, одно из которых *more* или *most* (more big ➔ bigger).
Преобразование существительного	В ошибке и исправлении ровно одно слово, и оба слова имеют одинаковую лемму, а также являются существительными. При этом слово в ошибке отсутствует в словаре (countrys ➔ countries).
Число существительного	В ошибке и исправлении ровно одно слово, и оба слова имеют одинаковую лемму, а также являются существительными (cat ➔ cats).
Притяжательный падеж существительного	Одно из слов в ошибке или исправлении имеет корректную часть речи, а также одно из слов заканчивается на ‘s (friends ➔ friend's).
Форма глагола	В ошибке и исправлении находятся глаголы в формах инфинитива, герундия или причастия (to eat ➔ eating).
Преобразование глагола	В ошибке и исправлении ровно одно слово, и оба слова имеют одинаковую лемму, а также являются глаголом, при этом слово в ошибке отсутствует в словаре (getted ➔ got).
Согласование с глаголом	Ошибка и исправления являются was и were. Ошибка и исправление содержат по одному слову, оба глаголы и хотя бы одно из них глагол в 3 форме от третьего лица (was ➔ were).
Время глагола	Целая группа, связанная с ошибками во времени глаголов, которые не подошли под предыдущие категории (has eaten ➔ was eating).
Остальное	Все остальные ошибки, которые не подошли ни под одну категорию.

Таблица 2. Типы грамматических ошибок

Тип	Форма
M – пропущено слово	Ø ➔ B
R – требуется замена	A ➔ B
U – лишнее слово	A ➔ Ø

Всего получилось 54 категории ошибок – прямое произведение типов и категорий за исключением тех, что не встречаются в исходных данных. Примеры из каждой категории были сгруппированы, причем ошибки, не относящиеся к данной категории, были исправлены, чтобы избежать их влияния на генерацию правил. Тем самым мы предполагаем, что в категории ошибок нет ни одной другой ошибки кроме как из данной категории.

2.3. Выделение признаков

После разделения данных по ошибкам из слов и предложения были выделены признаки (см. табл. 3) в качестве основы для генерации правил.

Таблица 3. Признаки на уровне слова

Признак	Описание
Лемма	Нормализованная, основная форма слова.
Частеречный тег (POS tag)	Один из 36 тегов из набора Penn Treebank (NN, JJ, VB, RB, …).
Тип словосочетания и место в нем	Тип словосочетания (chunk) согласно Penn Treebank и место в нем (начало, конец, середина).
Большая буква	Признак того, что слово начинается с большой буквы.
Начало/конец предложения	Признак того, что слово находится в начале или конце предложения.
Тип синтаксического отношения	Тип отношения согласно расширенным универсальным зависимостям (nsubj, obj, iojb, ...).

Пример размеченного предложения представлен на рисунке 2.

Рис. 2. Пример размеченного предложения

Для выделения признаков был использована библиотека StanfordNLP [20].

3. Генерация правил

3.1. Представление правила

Процесс проверки правописания выглядит следующим образом: предложение разбивается на токены, в нем выделяются признаки, описанные выше, после чего начинается итерирование по каждому токену. Токен подается в правило, как базовый элемент, от которого правило будет отталкиваться. В общем случае правило состоит из двух компонент: детектор и редактор. Детектор – это набор условий, которые должны быть выполнены в случае наличия ошибки. Условие – это тип признака вместе с ожидаемым его значением, либо логическая комбинация из других условий, а также смещение относительно проверяемого токена. Редактор – это набор изменений, которые должны произойти с предложением. Изменение состоит из:

— смещения относительно проверяемого токена;
— списка новых токенов, которые необходимо вставить перед изменяемым токеном;
— списка модификаций над изменяемым токеном (изменение регистра слова, изменение части речи слова, полная замена);
— флага о необходимости удаления токена.

Данный формат удобен для генерации правил в отличие от формата, принятого в LanguageTool. В нем правило выглядит похожим образом, поэтому трансляция из одного формата в другой при необходимости не вызывает сложностей (см. рис. 3).

```
<rule id="BED_ENGLISH" name="Possible typo 'bed/bat(bad) English/...'">
    <pattern>
      <marker>
        <token regexp="yes" negate="false">bed|bat</token>
      </marker>
      <token regexp="yes" postag="JJ|NN" postag_regexp="yes">English|attitude</token>
    </pattern>
    <message>Did you mean <suggestion>bad</suggestion>?</message>
    <example correction="bad" type="incorrect">Sorry for my <marker>bed</marker> English.</example>
</rule>
```

Рис. 3. Пример описания правила LanguageTool

<pattern> – это сгенерированное правило, внутри которого <token> является словом, на которое надо реагировать, в параметрах которого можно задавать признаки. <marker> отвечает за то, что будет являться ошибкой для пользователя и что нужно подчеркнуть. <example> – пример, берется из тестового набора данных. <suggestion> – это то, на что надо исправить <marker>, генерируется из редактора, внутри него можно использовать преобразования частей речи и обращаться к токенам в правиле по индексам. В итоге лингвисту остается только проверить осмысленность правила и придумать ему описание, хотя его тоже можно сгенерировать автоматически.

3.2. Transformation-Based Learning

В качестве метода генерации правил было использовано обучение на основе трансформаций [21]. Этот метод использовался, например, для генерации правил морфологической разметки. Процесс генерации правил выглядит следующим образом. На вход подается набор данных с ошибками и их исправлениями. По каждой ошибке генерируется правило для её детектирования согласно шаблону из заданного списка. К детектору добавляется редактор, который будет исправлять ошибку, т.е. набор изменений, которые необходимо сделать для исправления. На данном этапе мы получили список правил-кандидатов.

Теперь будем итерироваться по каждому кандидату и применять его ко всему набору данных, считая метрику качества. На каждой итерации выбираем лучшее правило по заданной метрике и применяем его ко всему набору данных. Повторяем, пока остаются ошибки, или результат не станет удовлетворительным (см. рис. 4).

```
while (кол-во ошибок > порог)
  foreach (ошибка)
    foreach (правило исправляющее ошибку)
       true_positive = кол-во правильно исправленных ошибок
         false_positive = кол-во неправильных срабатываний
         metric = true_positive - false_positive

  выбираем правило с самой большой metric и применяем его
  ко всем датасету и добавляем его в результат
```

Рис. 4. Псевдокод алгоритма TBL

Этот метод очень хорошо себя показал во многих областях обработки естественного языка [21]. Однако для применения метода к задаче исправления ошибок в тексте есть несколько препятствий:

— неясно, откуда брать список шаблонов для правил;
— подсчет метрики каждый раз для каждого правила занимает много времени;
— метрика плохо подходит для нашей задачи;
— неясно, какое правило считать лучшим на итерации.

Потенциально правилом может являться любая комбинация из признаков, а количество шаблонов очень быстро растет с ростом количества признаков и токенов. Если пересчитывать метрику каждый раз, то уже на 1000 правилах время выполнения будет достигать нескольких дней на персональном компьютере.

Если правило имеет большое количество одновременно ложных и корректных срабатываний, оно будет более приоритетным, в то время как более-менее продуктивное правило без ложных срабатываний будет выбрано гораздо позже или не выбрано вовсе. Помимо этого, возможна ситуация, когда два менее общих правила, которые будут покрывать то же множество ошибок, будут работать лучше и давать меньше ложных срабатываний. Применение более общего правила на более раннем этапе не позволит генерировать более специфичные правила позже.

Метод был взят за основу и модифицирован с учетом выявленных недостатков.

3.3. Предложенный метод

За основу была взята часть корпуса Гутенберга [22] размером в ~2 млн слов, без грамматических и прочих ошибок. Данные были разбиты на токены, выделены признаки. После чего был построен граф возможных последовательностей признаков длиной в три токена (далее тройка). На основе такой структуры можно быстро находить количество троек по заданному набору условий для трех подряд идущих токенов. Такие же графы были сгенерированы для всего набора данных и для троек, включающих ошибку из набора данных.

Для каждой ошибки в наборе генерируется каждая возможная тройка условий (порядка 250 тысяч на ошибку). Далее, тройки, которые часто (более 1% от всех троек) встречаются в «чистом» графе, а также тройки, которые реагируют на слишком малое количество (менее четырех) ошибок отфильтровываются, после чего выбирается первая тысяча лучших троек. После этого для каждой отфильтрованной тройки на основе графа из набора данных собираются индексы слов, на которые данная тройка будет реагировать, и рассчитывается количество правильных и ложных срабатываний.

В итоге остаются только те тройки, которые могут исправить более 3 ошибок и имеют менее 1000 ложных срабатываний.

Для каждой тройки с индексами генерируются условия для четвертого и пятого токена, получая тем самым наборы пятерок. Для каждой пятерки заново рассчитываются правильные и ложные срабатывания: при добавлении новых условий, количество срабатываний может только лишь уменьшится, следовательно, достаточно проверить только те срабатывания правила, которые уже нашлись для базовой тройки. Выбирается лучшая пятерка по заданной метрике для каждой ошибки. Если она имеет точность больше 80% и исправляет не меньше трех ошибок, то такая пятерка считается хорошей и из нее генерируется детектор.

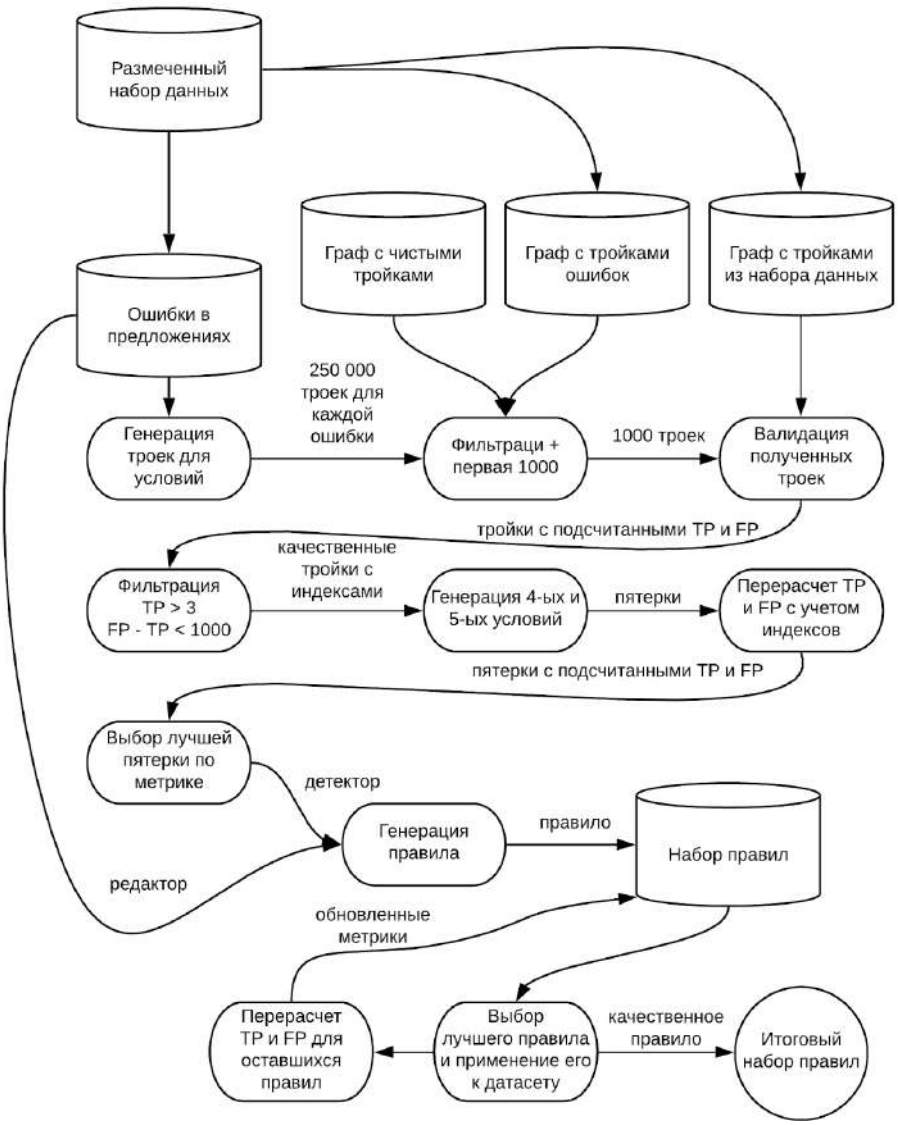

Рис. 5. Алгоритм работы предложенного метода генерации правил

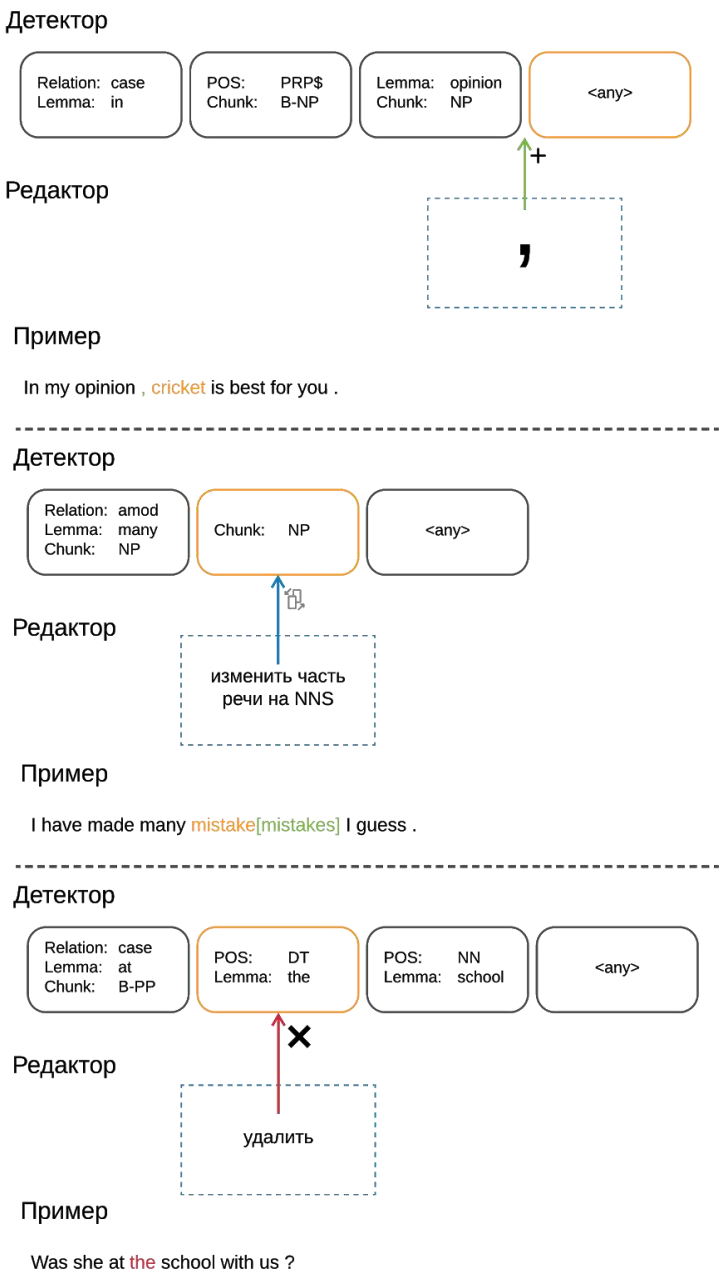

Рис. 6. Пример сгенерированных правил

Вместе с детектором единственно возможным образом из изначальной ошибки генерируется редактор. Из детектора и редактора получается правило-кандидат.

После прохождения по всем ошибкам и получения набора кандидатов в правила, дублирующие правила отфильтровываются. Каждый раз выбирается лучшее правило из набора и считается качественным, после чего выбранное правило применяется к набору данных. Затем выполняется перерасчет метрики для всех остальных правил из набора кандидатов и процесс повторяется до тех пор, пока очередное правило не перестанет удовлетворять заданной метрике.

Подробный процесс генерации правил показан на рисунке 5.

В качестве результата работы можно рассмотреть следующие правила (см. рис. 6). В первом примере правило можно сформулировать так: если фраза начинается с предлога *in*, далее идет любое притяжательное местоимение, за которым следует слово *opinion*, то после этой фразы следует поставить запятую. Во втором примере мы видим, что если встречается слово *many* являющееся часть связки с существительным, то существительное, к которому оно относится, должно находиться во множественном числе. Третье правило удаляет определенный артикль *the* во фразе *at the school*.

4. Оценка качества

Для оценки качества была использована F-мера, которая представляет собой гармоническое среднее между точностью и полнотой, она стремится к нулю, если точность или полнота стремятся к нулю. Стоит отметить, что при работе с правилами, большее значение придается точности, чем на полноте, потому что важнее правильно исправить ошибку, чем покрыть большее количество ошибок. В таких случаях F-мера может быть модифицирована с учетом необходимого приоритета. В настоящей работе приоритет точности считается в два раза выше приоритета полноты, такую меру принято называть $F_{0.5}$ (см. рис. 7).

$$F_{0.5} = (0.5^2 + 1)\frac{Precision \times Recall}{0.5^2 \times Precision + Recall}$$

Рис. 7. Формула расчет $F_{0.5}$-меры

4.1. Результаты генерации правил по категориям

Для части категорий с помощью разработанного метода были сгенерированы правила. Другая часть была исключена. Например, категории опечаток, орфографии, морфологии были исключены т.к. не рассматриваются в данной работе. Категории сокращений, преобразования существительного и глагола, формы прилагательного, были исключены ввиду слишком малого количества ошибок для обучения. Также была исключена категория “остальное”, потому что содержит в себе слишком разнородный набор ошибок. Для остальных категорий результаты представлены в табл. 4.

Таблица 4. Результаты генерации правил по категориям

Категория	Кол-во правил	TP	FP	FN	Precision	Recall	$F_{0.5}$
Пунктуация	523	3165	393	7483	0,89	0,30	0,64
Форма глагола	126	1164	130	3202	0,90	0,27	0,61
Согласование с глаголом	55	372	46	1011	0,89	0,27	0,61
Фразовые глаголы	12	156	23	422	0,87	0,27	0,60
Число существительного	72	394	21	2075	0,95	0,16	0,48
Артикли	184	1078	124	5915	0,90	0,15	0,46
Время глагола	58	277	20	3579	0,93	0,07	0,27
Предлоги	146	721	54	9694	0,93	0,07	0,27
Наречия	12	72	4	1195	0,95	0,06	0,23
Местоимения	23	100	10	1753	0,91	0,05	0,22
Существительные	11	56	5	3607	0,92	0,02	0,07
Глаголы	15	54	2	4193	0,96	0,01	0,06
Прилагательные	1	14	1	1270	0,93	0,01	0,05

После генерации правила были проверены на третьей части набора данных, которая не использовалась при генерации и показала следующие результаты (см. табл. 5):

Таблица 5. Результаты генерации правил по категориям

Категория	Кол-во правил	TP	FP	FN	Precision	Recall	$F_{0.5}$
Пунктуация	523	220	104	362	0.68	0.38	0.59
Фразовые глаголы	12	4	4	11	0.50	0.27	0.43
Наречия	12	5	3	34	0.63	0.13	0.35
Согласование с глаголом	55	18	19	97	0.49	0.16	0.34
Число существительного	72	25	23	180	0.52	0.12	0.31
Форма глагола	126	18	20	129	0.47	0.12	0.30
Артикли	184	44	38	404	0.54	0.10	0.28
Прилагательные	1	2	2	32	0.50	0.06	0.20
Предлоги	146	13	19	304	0.41	0.04	0.15
Существительные	11	2	1	81	0.67	0.02	0.11
Время глагола	58	3	14	182	0.18	0.02	0.06
Местоимения	23	1	4	103	0.20	0.01	0.04
Глаголы	15	0	2	169	0.00	0.00	0.00

4.2. Качество правил для LanguageTool

LanguageTool был запущен на данных для обучения, после чего правила, у которых было слишком много ложных срабатываний, были выключены. После чего к исходным правилам были добавлены правила, полученные с использованием предложенного подхода (см. табл. 6).

Таблица 6. Результаты LanguageTool

	TP	FP	FN	Precision	Recall	$F_{0.5}$
Сгенерированные правила	324	185	3699	0.67	0.08	0.27
LT	675	898	3224	0.43	0.17	0.33
LT + фильтр	632	494	3199	0.56	0.16	0.38
LT + фильтр + сгенерированные правила	905	665	3269	0.58	0.22	0.43

Выводы

С помощью предложенного метода удалось сгенерировать 1238 правил для 36 категорий. Качественнее всего правила получились для категорий пунктуации, формы глагола, согласования глагола с существительным и фразовых глаголах. Для категорий, в основном связанных с заменой частей речи, не удалось сгенерировать качественных и общих правил, т.к. ошибки в них сильнее зависят от контекста предложения, чем от ближайших 5 токенов к ошибке. Результаты на дополнительных данных это подтверждают, категория наречий оказалась впереди лишь по причине слишком малого количества ошибок этого типа в выборке.

Предложенный метод дает определенный прирост в качестве, полученные правила существенно расширили те 3203 правила, которые уже присутствуют в LanguageTool и увеличили количество найденных ошибок.

В дальнейшем планируется рассмотреть возможность генерации правил не для подряд идущих токенов в предложении, а для токенов, связанных через граф зависимостей, полученный после синтаксического анализа.

Это позволит сильнее учитывать контекст в рамках одного предложения и генерировать более общие правила для связок существительных с глаголом без учета лишних слов между ними.

Литература

[1] Bryant C., Felice M., Andersen Ø.E., Briscoe T. The BEA-2019 Shared Task on Grammatical Error Correction // Proceedings of the 14th Workshop on Innovative Use of NLP for Building Educational Applications. 2019. P. 52 – 75.
[2] Kumar A., Nair S. An artificial immune system-based approach for English grammar checking // de Castro L.N., Von Zuben F.J., Knidel H. (eds) Artificial Immune Systems. ICARIS 2007. Lecture Notes in Computer Science. 2007. Vol. 4628. P. 348 – 357.
[3] Huang A., Kuo T., Lai Y., Lin S. Identifying Correction Rules for Auto Editing // Proceedings of the 22nd Conference on Computational Linguistics and Speech Processing (ROCLING 2010). 2010. P. 251 – 165.
[4] Naber D. A Rule-Based Style and Grammar Checker. [Электронный ресурс] // Персональный сайт Даниэля Набера. 2003. URL: http://www.danielnaber.de/languagetool/download/style_and_grammar_checker.pdf (дата обращения: 16.05.2020).
[5] Soni M., Thakur J.S. A Systematic Review of Automated Grammar Checking in English Language. [Электронный ресурс] // arXiv.org. 2018. Дата обновления: 29.03.2018. URL: https://arxiv.org/pdf/1804.00540.pdf (дата обращения: 16.05.2020).
[6] Richter M., Stranák P., Korektor R.A. A system for contextual spell-checking and diacritics completion // Proceedings of the 24th International Conference on Computational Linguistics. 2012. P. 1019 – 1027.
[7] Kukich K. Techniques for automatically correcting words in text // ACM Computing Surveys. 1992. Vol. 24, № 4. P. 377 – 439.
[8] Dale R., Kilgarriff A. Helping Our Own: The HOO 2011 Pilot Shared Task // Proceedings of the 13th European Workshop on Natural Language Generation (ENLG). 2011. P. 242 – 249.
[9] Ng H.T., Wu S.M., Briscoe T., Hadiwinoto C., Susanto R.H., Bryant C. The CoNLL-2014 shared task on grammatical error correction // Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2014 Shared Task). 2014. P. 1 – 14.
[10] Rozovskaya A., Sammons M., Gioja J., Roth D. University of Illinois System in HOO Text Correction Shared Task // Proceedings of the 13th European Workshop on Natural Language Generation (ENLG). 2011. P. 263 – 266.
[11] Junczys-Dowmunt M., Grundkiewicz R. Phrase-based Machine Translation is State-of-the-Art for Automatic Grammatical Error Correction // Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. 2016. P. 1546 – 1556.
[12] Bryant C., Briscoe T. Language Model Based Grammatical Error Correction without Annotated Training Data // Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications. 2018. P. 247 – 253.
[13] Stahlberg F., Bryant C., Byrne B. Neural Grammatical Error Correction with Finite State Transducers // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2019. Vol. 1. P. 4033 – 4039.
[14] Zhao W., Wang L., Shen K., Jia R., Liu J. Improving Grammatical Error Correction via Pre-Training a Copy-Augmented Architecture with Unlabeled Data // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2019. Vol. 1. P. 156 – 165.
[15] Grundkiewicz R., Junczys-Dowmunt M., Heafield K. Neural Grammatical Error Correction Systems with Unsupervised Pre-training on Synthetic Data // Proceedings of the Fourteenth

Workshop on Innovative Use of NLP for Building Educational Applications. 2019. P. 252 – 263.
[16] Kantor Y., Katz Y., Choshen L., Cohen-Karlik E., Liberman N., Toledo A., Menczel A., Slonim N. Learning to combine Grammatical Error Corrections // Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications. 2019. P. 139 – 148.
[17] Yannakoudakis H., Briscoe T., Medlock B. A new dataset and method for automatically grading ESOL texts // Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. 2011. P. 180 – 189.
[18] Granger S. The computer learner corpus: A versatile new source of data for SLA research // Sylviane Granger, editor, Learner English on Computer. 1998. P. 3 – 18.
[19] Bryant C., Felice M., Briscoe T. Automatic annotation and evaluation of Error Types for Grammatical Error Correction // Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. 2017. Vol. 1. P. 793 – 805.
[20] Manning C.D., Surdeanu M., Bauer J., Finkel J., Bethard S.J., McClosky D. The Stanford CoreNLP Natural Language Processing Toolkit // Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. 2014. P. 55 – 60.
[21] Brill E. Transformation-Based Error-Driven Learning and Natural Language. A Case Study in Part of Speech Tagging // Computational Linguistics. 1995. Vol. 21, № 4. P. 543 – 565.
[22] Lahiri S. Complexity of Word Collocation Networks: A Preliminary Structural Analysis // Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics. 2014. P. 96 – 105.

Development and Implementation of Rule Generation Methods for Automatic Grammatical Error Correction

P. Bakhvalov

ITMO University

The task of automatic grammatical error correction (GEC) is still relevant. The amount of written text increases every year, as does the number of people starting to learn new languages, and it is common for all of us, as humans, to make mistakes.
There are two main approaches to solve this problem: an approach based on machine learning and a rule-based approach. Machine learning shows better results and does not require the involvement of professional linguists. The rule-based approach allows showing the user the description of the error based on language rule and requires significantly less computing resources. These approaches can be combined for quality increasing.
This paper presents a method of automatic obtaining of rules from an annotated dataset, so existing GEC-systems can be improved by generated rules after adding a description.
The Transformation-Based Learning (TBL) approach was taken as a basis, which was modified to use on data with a large number of features. As a result, 1238 rules for 36 categories of errors were generated. After that, the existing LanguageTool GEC-system was supplemented with the generated rules and showed significantly quality improvement.

Keywords: grammar check, automatic error correction, rule generation, rule-based, error detection

Reference for citation: Bakhvalov P. Development and implementation of rule generation methods for automatic grammatical error correction // Computer Linguistics and Computing Ontologies. Vol. 4 (Proceedings of the XXIII International Joint Scientific Conference «Internet and Modern Society», IMS-2020, St. Petersburg, June 17-20, 2020). - St. Petersburg: ITMO University, 2020. P. 83 – 97. DOI: 10.17586/0000-0000-2020-4-83-97

References

[1] Bryant C., Felice M., Andersen Ø.E., Briscoe T. The BEA-2019 Shared Task on Grammatical Error Correction // Proceedings of the 14th Workshop on Innovative Use of NLP for Building Educational Applications. 2019. P. 52 – 75.

[2] Kumar A., Nair S. An artificial immune system-based approach for English grammar checking // de Castro L.N., Von Zuben F.J., Knidel H. (eds) Artificial Immune Systems. ICARIS 2007. Lecture Notes in Computer Science. 2007. Vol. 4628. P. 348 – 357.

[3] Huang A., Kuo T., Lai Y., Lin S. Identifying Correction Rules for Auto Editing // Proceedings of the 22nd Conference on Computational Linguistics and Speech Processing (ROCLING 2010). 2010. P. 251 – 165.

[4] Naber D. A Rule-Based Style and Grammar Checker // Personal site of Daniel Naber. 2003. URL: http://www.danielnaber.de/ languagetool/download/style_and_grammar_checker.pdf (access date: 16.05.2020).

[5] Soni M., Thakur J.S. A Systematic Review of Automated Grammar Checking in English Language // arXiv preprint arXiv:1804.00540. 2018. URL: https://arxiv.org/pdf/1804.00540.pdf (access date: 16.05.2020).

[6] Richter M., Stranák P., R. A. Korektor A system for contextual spell-checking and diacritics completion // Proceedings of the 24th International Conference on Computational Linguistics. 2012. P. 1019 – 1027.

[7] Kukich K. Techniques for automatically correcting words in text // ACM Computing Surveys. 1992. Vol. 24, № 4. P. 377 – 439.

[8] Dale R., Kilgarriff A. Helping Our Own: The HOO 2011 Pilot Shared Task // Proceedings of the 13th European Workshop on Natural Language Generation (ENLG). 2011. P. 242 – 249.

[9] Ng H.T., Wu S.M., Briscoe T., Hadiwinoto C., Susanto R.H., Bryant C. The CoNLL-2014 shared task on grammatical error correction // Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2014 Shared Task). 2014. P. 1 – 14.

[10] Rozovskaya A., Sammons M., Gioja J., Roth D. University of Illinois System in HOO Text Correction Shared Task // Proceedings of the 13th European Workshop on Natural Language Generation (ENLG). 2011. P. 263 – 266.

[11] Junczys-Dowmunt M., Grundkiewicz R. Phrase-based Machine Translation is State-of-the-Art for Automatic Grammatical Error Correction // Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. 2016. P. 1546 – 1556.

[12] Bryant C., Briscoe T. Language Model Based Grammatical Error Correction without Annotated Training Data // Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications. 2018. P. 247 – 253.

[13] Stahlberg F., Bryant C., Byrne B. Neural Grammatical Error Correction with Finite State Transducers // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2019. Vol. 1. P. 4033 – 4039.

[14] Zhao W., Wang L., Shen K., Jia R., Liu J. Improving Grammatical Error Correction via Pre-Training a Copy-Augmented Architecture with Unlabeled Data // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2019. Vol. 1. P. 156 – 165.

[15] Grundkiewicz R., Junczys-Dowmunt M., Heafield K. Neural Grammatical Error Correction Systems with Unsupervised Pre-training on Synthetic Data // Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications. 2019. P. 252 – 263.

[16] Kantor Y., Katz Y., Choshen L., Cohen-Karlik E., Liberman N., Toledo A., Menczel A., Slonim N. Learning to combine Grammatical Error Corrections // Proceedings of the

Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications. 2019. P. 139 – 148.
[17] Yannakoudakis H., Briscoe T., Medlock B. A new dataset and method for automatically grading ESOL texts // Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. 2011. P. 180 – 189.
[18] Granger S. The computer learner corpus: A versatile new source of data for SLA research // Sylviane Granger, editor, Learner English on Computer. 1998. P. 3 – 18.
[19] Bryant C., Felice M., Briscoe T. Automatic annotation and evaluation of Error Types for Grammatical Error Correction // Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. 2017. Vol. 1. P. 793 – 805.
[20] Manning C.D., Surdeanu M., Bauer J., Finkel J., Bethard S.J., McClosky D. The Stanford CoreNLP Natural Language Processing Toolkit // Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. 2014. P. 55 – 60.
[21] Brill E. Transformation-Based Error-Driven Learning and Natural Language. A Case Study in Part of Speech Tagging // Computational Linguistics. 1995. Vol. 21, № 4. P. 543 – 565.
[22] Lahiri S. Complexity of Word Collocation Networks: A Preliminary Structural Analysis // Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics. 2014. P. 96 – 105.