Синтаксический анализ текста с орфографическими ошибками в системе Dictascope Syntax

Syntax parsing for texts with misspellings in dictascope syntax

Ерехинская Т. Н. (te@dictum.ru)

Титова А. С. (titova@dictum.ru)

Окатьев В. В. (oka@dictum.ru)

ООО «Диктум», Нижний Новгород, Россия

В статье рассматривается синтаксический анализ ЕЯ-текстов с опечатками. Предлагается способ интеграции модуля проверки орфографии и синтаксического анализатора, позволяющий с одной стороны исправлять опечатки с учетом контекста и повысить устойчивость синтаксического анализатора с другой.

Введение

Орфографические ошибки, опечатки, намеренное искажение слов — одна из характерных особенностей современного текстового контента. Борьба с опечатками давно вышла за рамки текстовых редакторов, где исправление проводилось в интересах человека — читателя. Такие небольшие недоразумения как пропущенная буква или вставленный по ошибке пробел мешают автоматической обработке текста, поэтому многие системы АОТ включают в себя исправление опечаток.

Поисковые системы вынуждены исправлять — «переколдовывать» — опечатки в запросах. Системам мониторинга блогов и форумов также требуется исправление опечаток или умение обходить их.

Многие прикладные приложения обработки текста включают синтаксический анализатор в качестве компонента. С одной стороны это позволяет добиться хорошего качества работы, с другой — делает приложение зависимым от результатов синтаксического анализа. Большинство синтаксических анализаторов не справляется с обработкой предложений, содержащих опечатки. Таким образом, само приложение также становится уязвимым.

Задача исправления орфографических ошибок имеет длинную историю [5–7,14]. Традиционные методы фокусировались на исправлении ошибок вставки, удаления, замены и перестановки символов в неизвестных словах

(то есть словах, которые не содержатся в используемом словаре). Heidorn [8] и Garside [9] разработали систему, которая полагалась на синтаксические шаблоны в распознавании ошибки замены, когда как Mays [10] использовали данные совместной встречаемости слов из большого корпуса, чтобы обнаружить и исправить такие ошибки [11]. Совместная встречаемость слов используется и в исправлении запросов в поисковой системе Yandex [12].

Некоторые системы проверки орфографии (например, ОРФО) рассматривают только одноместные замены внутри неизвестного слова. Это значит, что слова, в написании которых допущено более одной ошибки, останутся без вариантов замены.

Наиболее близкий подход можно найти в статье Vosse [15], описывающей интеграцию модуля проверки орфографии и синтаксического анализатора для немецкого языка. Особое внимание автор уделил ошибкам согласования, в результате которых отдельные слова в предложении остаются словарными, но само предложение перестает быть грамматически верным. В работе использован анализатор на основе алгоритма Томиты. Для обработки структурных ошибок в грамматику были введены дополнительные правило.

По сравнению с работой Vosse, целью данной работы является повышение устойчивости анализатора к опечаткам за счет минимальных изменений.

1. Постановка задачи

Задачу синтаксического анализ текста с орфографическими ошибками можно рассматривать с двух сторон. Во-первых, в синтаксическом анализаторе возможно использование модуля, предлагающего варианты исправлений для слов с ошибками. Это позволит повысить устойчивость синтаксического анализатора. Во-вторых, учет синтаксического контекста может быть очень полезен для корректного исправления опечаток. Например, в предложении «*мне нравиться телефон*» все слова написаны правильно, однако само предложение некорректно.

Цель данной работы — расширение возможностей синтаксического анализатора за счет внедрения исправления опечаток и совершенствование исправления опечаток за счет использования синтаксического анализа. Схема взаимодействия модулей проиллюстрирована на рис. 1.

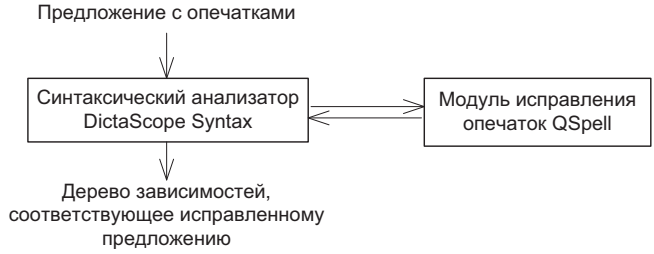

Рис. 1. Взаимодействие модулей синтаксического анализа и исправления опечаток

2. Виды опечаток

При исправлении орфографии должны учитываться следующие типы ошибок:

1. вставка лишнего символа *«будующее — будущее»*;
2. пропуск символа *«грусно — грустно»*;
3. замена одного символа на другой *«кинтакт — контакт»*;
4. транспозиция рядом стоящих символов *«рбаво — браво»*.

Также надо учитывать особенности печати слов на клавиатуре и типичные ошибки. К ним можно причислить:

1. клавиатурная близость клавиш: *«анеудот — анекдот»*;
2. ошибки в безударных гласных: *«аностасия — анастасия»*;
3. фонетическая похожесть букв: *«брюпетка — брюнетка»*;
4. парные буквы: *«автограв — автограф»*;
5. вставка лишнего пробела: *«сло во — слово»*;
6. отсутствие пробела или дефиса: *«футбольныйклуб — футбольный клуб»*;
7. идентичное написание букв в разных раскладках: «хромосом — хромосом»;
8. схожесть написания цифр и букв (ч-4, о-0, з-3): *«4естно — честно»*;
9. буквы и символы в разных раскладках: *«<лизнец — близнец»*;
10. ошибки после шипящих и ц: *«жолтый — желтый»*;
11. перепутывание и смещение рук при слепой печати: *«инвнжае — телефон»*;
12. перевод транслитерации на русское написание: *«kartinki — картинки»*;
13. исправление неправильной раскладки клавиатуры как для целого, так и для части слова: *«jlyjrkfccybrb — одноклассники»*.

3. Исправление опечаток

3.1. Базовый алгоритм

Постановка задачи исправления ошибок и опечаток в словах может быть сформулирована следующим образом:

Пусть $\sum$ будет алфавитом нашего языка и $L \subset \sum^*$ — словарь грамматически правильных слов. Для слова $w' \in \sum^* \setminus L$ требуется найти слова $w' \in L$ такие, что $dist(w,w') \le \delta$ и $F(w') = \max\limits_{v \in L: dist(w,v) \le \delta} F(v)$. [11]

Для некоторого несловарного слова из текста требуется найти одну или несколько ближайших словоформ из словаря, и выбрать из них наиболее частотные, где:

$dist(w,w')$ — расстояние между двумя словами – мера, показывающая насколько одно слово похоже на другое. Будем использовать модифицированный подсчет расстояния Левенштейна.

δ — пороговое значение расстояния. Если $dist(w, w') \leq \delta$, то считаем, что слова w и w'близки, то есть, можно сказать, что слово w' является исправлением несловарного слова w. $F(w)$ — частота употребления слова w.

В классическом варианте алгоритма Левенштейна вес операций устанавливают равным единице. Модификация заключается в том, что каждой паре символов, участвующей в определении расстояния Левенштейна, назначен свой вес из интервала . Если нужный вес не находится среди данных матриц, то он автоматически считается равным единице. Для операций определены следующие матрицы весов:

1. замена одного символа на другой;
2. замена символов после шипящих и ц;
3. транспозиция рядом стоящих символов;
4. вставка символов;
5. удаление символов.

Подобная модификация объясняется тем, что буквы определенного алфавита используются неравномерно: некоторые символы употребляются чаще других, поэтому участвуют в большем количестве ошибок. Помимо этого, матрицы позволяют учесть особенности печати на клавиатуре (описка *«а-п»* более вероятная, чем *«а-ч»* в силу расположения клавиш на клавиатуре), назначить свой собственный вес для фонетически похожих символов (ошибка *«в-ф»* будет более частотной, следовательно, и более вероятной, чем *«г-к»*), распределить соответствие символов русского алфавита, латиницы, цифр и знаков (*«т-м»* и *«т-т»*).

3.2. Лексический контекст

У отдельного слова с опечаткой может быть несколько вариантов исправления, причем исправления могут иметь разные грамматические характеристики.

Пример. *гоод*

Варианты исправления: «*год*», «*город*», «*голод*», «*горд*», «*голд*».

Как видно из примера, при исправлении опечаток недостаточно близости символьного представления слов.

Пример. *аренда катра*

В данном случае «*катра*» может быть «*катером*» и «*картой*», но по окружению слов легко можно понять, что имеется в виду «*катер*».

При исправлении такого рода ошибок следует учитывать контекст, поскольку при неправильном выборе варианта исправления может измениться

или потеряться смысл (*белый грип — белый гриб*, а *птичий грип — птичий грипп*). В этих целях используется словарь биграмм или словарь сочетаемости слов.

Данный словарь используется также и при исправлении пропуска пробела, когда два слова ошибочно написаны слитно.

Пример. *футбольныйклуб — футбольный клуб*

Считается, что от 80 % до 90 % всех опечаток содержат одну ошибку в слове и орфографический корректор, пытающийся достичь 90 % точности, обязательно должен использовать контекст [13].

3.3. Словарное слово + опечатка = словарное слово

До этого момента мы рассматривали исправление ошибок только среди несловарных слов. Однако то, что рассматриваемое слово содержится в словаре, еще не означает, что оно не содержит ошибок. Довольно распространенный тип ошибок — это ошибки в окончаниях глаголов на -тся/-ться.

Пример. *очень нравиться телефон*

Слово «*нравиться*» является словарным, но оно также содержит ошибку.

Если такая ошибка не будет исправлена, то синтаксический анализатор не сможет корректно построить дерево синтаксического разбора, поскольку будет отсутствовать связь между словами «*нравиться*» и «*телефон*». Само слово является словарным, поэтому на этапе исправления нельзя отбрасывать исходное слово. Модуль исправления опечаток предложит варианты исправления, а синтаксический анализатор сделает окончательный выбор.

3.4. Ошибки в окончаниях

Наверное, каждый из нас сталкивался с тем, что в ходе написания фразы мысленно ее переформулировал, после чего появляются ошибки в согласовании и управлении слов. Предположим, что хотели написать предложение «*Он был веселым мальчиком*», в процессе набора решили перефразировать его до «*Он был веселый мальчик*», а в результате получили «*Он был веселым мальчик*». Чтобы исправить ошибку согласования, требуется прилагательное поставить в нужный падеж, для чего необходимо подобрать правильное окончание. Для этого на этапе исправления искажений получаем возможные формы слова «*веселый*», затем синтаксический анализатор выберет нужную форму.

4. Синтаксис

4.1. Принятая модель

Синтаксический анализатор DictaScope строит дерево зависимостей для предложения [2–4]. Реализованный подход основан на следующей модели.

Пусть на вход анализатору приходит цепочка слов S: $S = w_1, w_2, ..., w_N$, где w_i — лексема при $i = \overline{1, N-1}$, w_N — слово, соответствующее фиктивному корню дерева зависимостей. По входной цепочке слов строится ориентированный взвешенный граф $G = \langle V, E \rangle$, вершинам графа назначены номера: $V = \{1, ..., n\}$. Задано разбиение V на отрезки-классы $I_k = [a_k, b_k]$, $k = \overline{1, N}$, $a_1 = 1$, $a_{k+1} = b_k + 1$, $k = \overline{1, N-1}$, $w_N = w_N = n$. Каждое множество I_k соответствует множеству омоформ слова w_k в предложении. Под омоформой понимается пара: грамматическое значение и начальная форма.

Для построения дерева зависимостей используется модифицированный алгоритм Эйснера [1]. Модификация включает проверку пунктуации и некоторые дополнительные лингвистические ограничения. Однако для целей данной работы важным является лишь то, что одному слову соответствует несколько вершин-интерпретаций.

Далее приведен модифицированный алгоритм Эйснера. Обозначим номер множества, содержащего вершину $j \in V$: $ind(j) \stackrel{def}{=} k \Leftrightarrow j \in I_k$. Введем функцию расстояния между двумя вершинами $L(r, l)$, $r, j \in V$.

$L(r, l) = |ind(r) - ind(l)| - c$,
где c — количество сочинительных союзов между словами $ind(r)$ и $ind(l)$.

Входом алгоритма является описанный выше граф $G = \langle V, E \rangle$ выходом - минимальное проективное дерево $\dot{O} = \langle V^*, E^* \rangle$: $V^* \subseteq V$, $E^* \subseteq E$, причем из каждого класса в дерево входит строго одна омоформа: $\forall k \in \{1, ..., N\}$ $|I_k \cap V^*| = 1$.

Алгоритм Эйснера представляет из себя заполнение таблицы $C[x][y][d][q]$, где x, y — левая и правая вершины, d — направление поддерева, q — показатель завершенности.

Начало алгоритма
C[s][s][d][c] = 0 ∀s; d; c
для k: 1..n
    для s: 1..n
        t = s + k
        если t > n прервать цикл
        если *ind*(*s*) = *ind*(*t*)
            C[s][t][d][c] = 0 ∀d; c
            следующая итерация

$$C[s][t][\leftarrow][0] = \min_{\substack{s\le r,l\le t \\ L(r,l)=1}} \left(C[s][r][\rightarrow][1] + C[l][t][\leftarrow][1] + s(t;s)\right)$$

$$C[s][t][\rightarrow][0] = \min_{\substack{s\le r,l\le t \\ L(r,l)=1}} \left(C[s][r][\rightarrow][1] + C[l][t][\leftarrow][1] + s(s;t)\right)$$

$$C[s][t][\leftarrow][1] = \min_{\substack{s\le r<t \\ \mathrm{ind}(t)\neq\mathrm{ind}(r)}} \left(C[s][r][\leftarrow][1] + C[r][t][\leftarrow][0]\right)$$

$$C[s][t][\rightarrow][1] = \min_{\substack{s<r\le t \\ \mathrm{ind}(t)\neq\mathrm{ind}(r)}} \left(C[s][r][\rightarrow][0] + C[r][t][\rightarrow][1]\right)$$

конец цикла
конец цикла
конец алгоритма

Решению соответствует клетка матрицы
$C[0][k^*][x^*][y^*] : k^*, x^*, y^* = \arg\min_{k,x,y} C[0][k][x][y]$.

Для получения результирующего дерева выполняется проход сверху вниз. Для этого нужно поддерживать обратные указатели на поддеревья, которые составляют каждый элемент таблицы.

4.2. Адаптация модели для исправления опечаток

Обсуждение интеграции исправления опечаток и синтаксического анализа начнем с наиболее простого случая — исправление одного слова на другое.

Пусть слову w_k соответствуют варианты исправления $w_1^k, w_2^k, \ldots, w_Q^k$. Тогда расширим множество омоформ I_k, добавив к нему омоформы, соответствующие вариантам исправления: $I_k = [a_k, b_k] \cup [\tilde{a}_1^k, \tilde{b}_1^k] \cup [\tilde{a}_2^k, \tilde{b}_2^k] \cup \ldots \cup [\tilde{a}_Q^k, \tilde{b}_Q^k]$, сохранив принцип нумерации: $\tilde{a}_1^k = b_k + 1$, $a_{k+1} = \tilde{b}_Q^k + 1$, $\tilde{a}_{i+1}^k = \tilde{b}_i^k + 1$, $k = \overline{1, N-1}$. Кроме того, расширим представление омоформы до тройки: вариант исправления (может совпадать с исходным словом), грамматическое значение, начальная форма.

Таким образом, множество омоформ слова w_k расширено за счет добавления омоформ вариантов его исправлений – это соответствует добавлению новых вершин в граф *G*. Потенциальные связи для новых вершин устанавливаются на общих основаниях с «обычными» вершинами – по базе синтаксических правил.

Пример. *Мне нравиться телфон.*

Слово *телфон* может быть проинтерпретировано как несловарное слово, в этом случае ему будут соответствовать 6 грамматических значений имени собственного (во всех падежах). Модуль исправления опечаток предложит два варианта: *телефон* и *тефлон*. Каждому соответствует по две омоформы — в именительном и винительном падеже. Таким образом, слову *телфон* будет соответствовать 10 вершин в графе *G*. Аналогично, для слова *нравиться* будет предложено исправление *нравится*.

В качестве результирующего может быть выбрано дерево для следующих вариантов: *«мне нравится телефон»* и *«мне нравится тефлон»*. Кроме того, возможное решение — вариант *«мне нравится телфон»*, где для слова *телфон* возможны грамматические значения имени собственного в именительном, дательном или творительном падежах. Однако вес деревьев со словом *телфон* содержит высокий штраф за несловарное слово, остаются только деревья с исправлениями. Учет частоты встречаемости слов помогает выбрать между *телефоном* и *тефлоном*.

Таким образом, в рамках принятой модели задача исправления опечаток в отдельном слове решается за счет дополнительных омоформ. Однако исправление опечаток не ограничивается коррекцией отдельного слова. Более сложный случай — разделение одного слова на два или соединение двух слов в одно. Проведем дальнейшую модификацию алгоритма.

По входной цепочке слов $S = w_1, w_2, ..., w_N$ модуль исправления опечаток может предложить варианты склейки/разделения слов: w_i, $w_{i+1} \to \overline{w}_i$ или $w_i \to w_{i1}, w_{i2}$. Как включить использование этой информации в алгоритм Эйснера?

Суть алгоритма Эйснера — построить наилучшие деревья зависимостей на отрезках, а затем собрать из них результирующее дерево.

При построении дерева на отрезке $[w_i, w_i]$можно рассматривать все слова с их омоформами, а можно исключить из рассмотрения одно или несколько слов, как это показано на рис. 2.

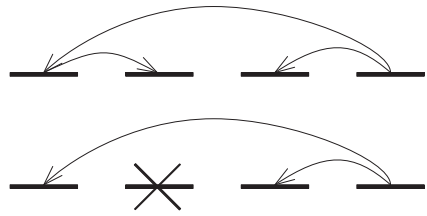

Рис. 2. Построение дерева на отрезке с исключением слова

Будем говорить, что слово, которое при некоторых условиях может не попасть в результирующее дерево, является опциональным. В контексте текущей задачи важно понять, что же мы считаем словом. На самом деле слово можно рассматривать как некоторую абстракцию, содержащую набор омоформ. Слова задают классы в графе *G*, способ нумерации вершин и ограничение на связи — вершины из одного класса не могут быть связаны.

Возьмем начальное разбиение $S = w_1, w_2, ..., w_N$, построим множество вершин *V*. Затем для каждого варианта исправления выполним следующие действия. Для исправлений вида $w_i \to w_{i1}$, w_{i2} добавим к омоформам слова w_i омоформы слова w_{i1}, добавим опциональное слово w_{i2} в цепочку *S* после слова w_i и омоформы слова w_{i2} в множество вершин *V*. Если для слова w_i есть несколько вариантов исправления вида $w_i \to w_{i1}$, w_{i2}, все омофрмы для различных вариантов w_{i2} приписываются новому опциональному слову.

Для исправлений вида $w_i, w_{i+1} \to \overline{w}_i$ к омоформам слова w_i добавляются омоформы слова $\overline{w}_i$, слово w_{i+1} помечается как опциональное.

После выполнения этих операций для всех вариантов исправления необходимо перенумеровать слова в S, вершины в G и слова в вариантах исправлений. Кроме того, при добавлении вершин необходимо сохранять информацию о несовместимости вершин из соседних слов. Если две вершины несовместимы, в дерево может попасть не более одной из них. Проиллюстрируем сказанное примером.

Пример. *Лучшепотом.*

$S = w_1$, $w_1 \to$ *«Лучшепотом»*. Варианты исправления: $w_1 \to$ *«Лучше», «потом»*, $w_1 \to$ *«Луч», «шепотом»*. С учетом исправлений $S = w_1, w_2$, w_2 — опциональное слово. Множество вершин V показано на рис. 3. Символ ∅ соответствует исключению опционального слова из рассмотрения. Пунктиром показаны несовместимые вершины.

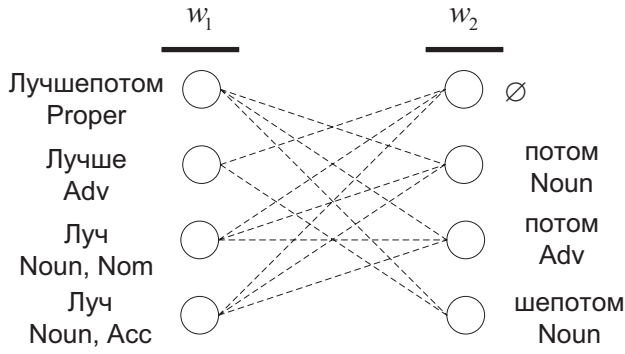

Рис. 3. Множество вершин для учета исправлений

Дополнительно в каждой ячейке таблицы $C[x][y][d][q]$ будем хранить список вершин, несовместимых с деревом, соответствующим данной ячейке. Для учета совместимости вершин при заполнении ячейки поиск минимального дерева ведется только среди тех деревьев, в которых все вершины попарно совместимы.

Заключение

Предложенный способ интеграции синтаксического анализатора и модуля исправления опечаток был реализован в системе DictaScope Syntax. Это позволило существенно повысить устойчивость анализатора и дало возможность использовать его в качестве компонента в системе извлечения мнений для мониторинга блогов и форумов.

Была проведена серия экспериментов с целью получения численной оценки качества анализа на текстах с опечатками. Полнота и точность модуля исправления ошибок и опечаток в словах составили 85.0% и 80.8% соответственно,

скорость — 0.15 Кб/с. Качество синтаксического анализа (процент правильно построенных деревьев синтаксических связей) на текстах с опечатками выросло с 23 % до 65 %. Данные были получены на машине Athlon 3,1 GHz.

Литература

1. *Eisner J.* (1996) Three new probabilistic models for dependency parsing: An exploration. In: Proceedings of the 16th International Conference on Computational Linguistics (COLING), pp 340–345
2. *Окатьев В. В., Ерехинская Т. Н., Скатов Д. С.* Модели и методы учета пунктуации при синтаксическом анализе предложения русского языка // Труды Международной конференции Диалог'2009.
3. *Окатьев В. В., Ерехинская Т. Н., Ратанова Т. Е.* Тайные знаки пунктуации. // Труды Международной конференции Диалог'2010.
4. *Окатьев В. В., Гергель В. П., Алексеев В. Е., Таланов В. А., Баркалов К. А., Скатов Д. С., Ерехинская Т. Н., Котов А. Е., Титова А. С.* Отчет о выполнении НИОКР по теме: «Разработка пилотной версии системы синтаксического анализа русского языка» (инвентарный номер ВНТИЦ 02200803750) // М.: ВНИТЦ, 2008
5. *Damerau, F. J.* 1964. A technique for computer detection and correction of spelling errors. In Communications of ACM, 7(3):171–176.
6. *Rieseman, E. M. and A. R. Hanson.* 1974. A contextual postprocessing system for error correction using binary n-grams. In IEEE Transactions on Computers, 23(5) : 480–493.
7. *McIlroy, M. D.* 1982. Development of a spelling list. In JIEEE-TRANS-COMM, 30(1); 91–99
8. *Heidorn, G. E., K. Jensen, L. A. Miller, R. J. Byrd and M. S. Chodorow.* 1982. The EPISTLE text-critiquing system. In IBM Systems Journal, 21(3):305–326.
9. *Garside, R., G. Leech and G. Sampson.* 1987. Computational analysis of English: a corpus-based approach, Longman.
10. *Mays, E., F. J. Damerau and R. L. Mercer.* 1991. Contextbased spelling correction. In Information Processing and Management, 27(5) : 517–522.
11. *S. Cucerzan, and E. Brill.* Spelling correction as an iterative process that exploits the collective knowledge of web users. Microsoft Research. One Microsoft Way. Redmond, WA 98052
12. *Байтин А.* Исправление поисковых запросов в Яндексе. Вероятностная языковая модель. 2008.
13. *Peter Norvig* «How to Write a Spelling Corrector». http://norvig.com/spell-correct.html
14. *Eric Sven Ristad, Peter N. Yianilos.* «Learning String-Edit Distance». IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 5, May 1998
15. *Theo Vosse.* Detecting and correcting morpho-syntactic errors in real texts, Proceedings of the third conference on Applied natural language processing, March 31-April 03, 1992, Trento, Italy.