Московский Государственный Университет имени М.В.Ломоносова
Факультет Вычислительной Математики и Кибернетики
Кафедра Системного Программирования

Выпускная квалификационная работа

"Исправление орфографических ошибок в русскоязычных отзывах в сети Интернет"

Автор:
гр. 427
Кириллов Алексей Павлович

Научный руководитель:
д-р техн. наук, профессор Кузнецов Сергей Дмитриевич

Москва, 2016

Содержание

1 Аннотация **4**

2 Введение **5**

3 Постановка задачи **7**

4 Обзор существующих решений **9**

4.1 Исправление ошибок на уровне слов 9

4.1.1 Множества кандидатов 9

4.1.2 Модели ошибок 11

4.1.3 Машинный перевод 12

4.2 Исправление ошибок на символьном уровне 13

4.3 Выводы 16

5 Исследование и построение решения задачи **17**

5.1 Базовый метод 17

5.2 Общая схема реализованного метода 17

5.3 Выявление ошибок 18

5.4 Предобработка некорректных слов 19

5.5 Генерация кандидатов для замены 20

5.5.1 Ближайшие по расстоянию Дамерау-Левенштейна 21

5.5.2 Устранение излишних повторений букв и слогов 23

5.5.3 Простые фонетические замены 24

5.6 Выбор оптимальной замены 25

5.6.1 Отбор слов с наименьшей стоимостью преобразования. 25

5.6.2 Выбор оптимальной замены на основе n-граммной модели языка . 25

5.7 Полная схема работы реализованного метода 26

5.8 Описание данных для обучения и тестирования 26

5.9 Способ и результаты тестирования 29

6 Описание практической части **31**

6.1 Обоснование выбранного инструментария 31

6.2 Схема работы программной системы 31

6.3 Общая архитектура 34

7 Заключение **36**

1 Аннотация

Данная выпускная квалификационная работа посвящена исследованию методов исправления орфографических ошибок в текстах на естественных языках и разработке и реализации метода, решающего задачу автоматического исправления ошибок в текстах на русском языке. В работе рассматриваются существующие методы исправления ошибок в текстах на естественных языках. Разработанный метод последовательно находит ошибки, генерирует варианты их исправления и выбирает наиболее соответствующего контексту кандидата для каждой из найденых ошибок. Поиск ошибок происходит методом просмотра словаря. Генерация вариантов — на основании обобщённого расстояния Дамерау-Левенштейна, предположений об излишних повторениях букв и слогов, а также информации о типичных фонетических ошибках.

2 Введение

Задача поиска и исправления ошибок в текстах (Spelling correction) является одной из наиболее базовых задач анализа текстов на естественных языках (Natural Language Processing). История попыток решить эту задачу берёт своё начало с середины шестидесятых годов предыдущего столетия с работ Владимира Иосифовича Левеншейна[6] и Фредерика Дамерау[5]. И вот уже на протяжении более полувека задача не теряет своей актуальности, появляются новые методы, расширяется область её применения. Твиттер и другие микробоговые сервисы очень привлекательны с точки зрения извлечения и анализа информации, содержащейся в сообщениях, так как информация появляется в режиме реального времени. Тем не менее текст в сети интернет часто очень сильно отличается от общепринятых норм языка. В нём могут содержаться ошибки, намеренные искажения слов, опечатки, а также большое число неологизмов и окказионализмов.

За 50 лет решения данной задачи исследователями было перепробовано огромное количество различных методов. От кодов символов и таблиц допустимости n-грамм и непосредственного применения расстояния Дамерау-Левенштейна до активного использования различных методов машинного обучения, фонетической информации о слове и методов машинного перевода.

Существуют различный способы классификации ошибок. Ошибки в текстах на естественном языке можно разделить на два класса: ошибки, в результате которых возникают некорректные слова и ошибки, в которых такие слова не возникают. В данной работе будут исправляться только ошибки первого типа, поскольку применяемый метод не способен обнаружить ошибки второго. В рамках данной работы будем выделять опечатки (например, искючительно/исключительно), орфографические (симпотичный/симпатчный) и грамматические ошибки (ихнего), намеренные искажения слов (ищо, урааа), ошибки в неправильной расстановке пробелов и дефисов (изза, илишестидесяти).При этом в рамках данной работы не считается ошибками употребление разговорных слов, и не учитываются ошибки в словах набранных латиницей, пунктуационные ошибки и ошибки капитализации.

В этой работе будут рассмотрены современные методы обнаружения и исправления ошибок в текстах, разработан метод решающий данную задачу, реализовано соответствующее программное средство, и проведён сравнительный анализ результатов их работы применительно к предложениям русского языка.

3 Постановка задачи

Будем считать словом непустую конечную последовательность букв русского и латинского алфавитов, а также цифр и знаков тире, т.е. $word \in$ {а,б,...,я,a,b,...,z,0,1,...,9,-}$^+$. А *Words* — множество всех слов. Пусть тогда предложение — это упорядоченная последовательность слов $S = \{w_1, w_2, ..., w_n\},\ w_i \in Words,\ i = \overline{1,n}$.

Пусть у нас есть два предложения $S_i = \{w_{i,1}, w_{i,2}, ..., w_{i,n_i}\}, i = \overline{1,2}$. Разобьём каждое из этих предложений на m, возможно пустых, отрезков $s_{i,j} = \{w_{i,k_{i,j-1}}, ..., w_{i,k_{i,j}-1}\}$, где $1 = k_{i,0} \leqslant k_{i,1} \leqslant ... \leqslant k_{i,m} = n_i, i = \overline{1,2}, j = \overline{1,m}$. Таким образом, в соответствие каждому из полученных подмножеств слов первого предложения $\forall j = \overline{1,m}\ s_{1,j}$ мы можем поставить подмножество слов второго предложения $s_{2,j}$, каждое из которых может быть, вообще говоря, пустым. В таком случае будем говорить, что между предложениями установлено соответствие, а пару $s_{1,j}$ и $s_{2,j}$ будем называется заменой.

Пусть j-ые отрезки в обоих предложениях содержат ровно по одному слову, причём это одно и то же слово $k_{1,j} - k_{1,j-1} = k_{2,j} - k_{2,j-1} = 1$ и $w_{1,k_{1,j-1}} = w_{2,k_{2,j-1}}$. В таком случае будем называть замену тождественной. Оптимальным будем называть такое соответствие предложений, при котором количество тождественных замен максимально. Нетождественные замены в оптимальном соответствии предложений S_1 и S_2 будем называть исправлениями ошибок в предложении S_1 при золотом стандарте S_2. При этом отрезок $s_{1,j}$ в каждой такой замене — это ошибка, а $s_{2,j}$ это её исправление.

Каждому предложению S соответствует его корректный аналог $S_{\text{корр}}$, определённый экспертами. Требуется разработать метод, исправляющий ошибки в S при золотом стандарте $S_{\text{корр}}$.

Например, пусть S = {'а', 'теперь', 'смотрите', 'чтоже', 'мы', 'увидили'}, а его корректный аналог $S_{\text{корр}}$ = {'а', 'теперь', 'смотрите', 'что', 'же', 'мы', 'увидели'}. Существует много вариантов установить соответствие между ними. Рассмотрим разбиения этих предложений вида {('а'), ('теперь'), ('смотрите'), ('чтоже'), ('мы'), ('увидили')} и {('а'), ('теперь'), ('смотрите'), ('что', 'же'), ('мы'), ('увидели')}. Сопоставление предложений, основанное на таком разбиении, получается четыре тождественных замены: ('а') $\leftrightarrow$ ('а'), ('теперь') $\leftrightarrow$ ('теперь'), ('смотрите') $\leftrightarrow$ ('смотрите'), ('мы') $\leftrightarrow$ ('мы'). И две не тождественные. Заметим, что увеличить количество тождественных замен в предложении нельзя. Следовательно установленное соответствие является оптимальным, а замены ('чтоже') $\leftrightarrow$ ('что', 'же') и ('увидили') $\leftrightarrow$ ('увидели') — исправления ошибок

в предложении S при золотом стандарте S. Причём ('чтоже') и ('увидили') являются ошибками, а ('что', 'же') и ('увидели') их исправлениями.

Для достижения поставленной цели, необходимо решить следующие задачи:

- проанализировать современные методы обнаружения и исправления ошибок в текстах на естественных языках;
- разработать метод исправления ошибок в предложениях применимый к данной задаче;
- спроектировать и построить программное средство, отвечающее цели данной работы;
- провести сравнительный анализ результатов работы реализованного средства с базовым методом решения задачи.

4 Обзор существующих решений

Существует два метода обнаружение ошибок: на основе словарей или на основе символьных n-грамм. В первом случае используется словарь правильных слов или лемм, и все слова текста, не входящие в этот словарь, считаются некорректными (в рамках данной работы не рассматриваются ошибки сочетаемости слов). Во втором случае происходит поиск не некорректных слов, а нестандартных символьных последовательностей. Этот метод, в отличие от словарного, может исправлять ошибки в новых словах, но при этом нередко ошибочно исправляет заведомо корректные слова. Эти два метода обнаружения ошибок порождают два класса методов исправления ошибок: методы рассматривающие предложение как последовательность слов или как последовательность символов.

4.1 Исправление ошибок на уровне слов

4.1.1 Множества кандидатов

Идея метода состоит в генерации множества кандидатов для каждого несловарного слова и последующего выбора наиболее подходящей замены. Метод нашёл применение, например, в статье Хана и Болдуина [1]. В работе рассматривается задача нормализации SMS-сообщений и твитов. Под нормализацией понимается приведение несловарных слов к некоторой стандартной форме. В рамках этой статьи рассматриваются только ошибки в несловарных словах, и производятся лишь замены токена на токен. Например, замена 'imo' на корректный аналог 'in my opinion' в этой работе не происходит.

Слова исходного сообщения, не найденные в словаре, могут представлять из себя слова с опечатками и орфографическими ошибками, а также имена собственные и неологизмы. Поэтому прежде чем исправлять такие слова, необходимо выяснить, содержит ли слово ошибку. С этой целью задача разбита на три этапа. Сначала происходит генерация множеств кандидатов для всех несловарных слов. На втором этапе происходит выявление некорректных слов, т.е. слов, которые в дальнейшем мы будем пытаться исправить. Наконец, на заключительном этапе происходит выбор наиболее подходящей замены для каждого из некорректных слов.

Генерация множеств кандидатов. Сначала происходит замена всех последовательностей из более чем трёх одинаковых букв на последовательности из трёх таких

букв. Например, слово 'cooool' будет заменено на 'coool'. Затем из словаря выбираются слова, находящиеся на расстоянии не более заданного числа (в статье исследуются значения от 0 до 2) от исходного в смысле Дамерау-Левенштейна. И слова, фонетическая транскрипция которых близка по этому же расстоянию к транскрипции исходного слова. Это необходимо для исправления ошибок типа 'earthquick'→'eartquake'. Увеличение максимально допустимого расстояния позволяет включить корректный аналог в множества кандидатов для большего количества некорректных слов, но при этом сильно возрастает размер этих множеств. Поэтому авторы статьи в итоге остановились на ограничении в два изменения.

Выявление некорректных слов. Предварительно из текстов газеты Нью-Йорк Таймс с помощью стенфордского морфологического парсера извлекаются зависимости. И запоминаются в виде троек (главное слово, зависимое, расстояние между ними). Например, для предложения 'One obvious difference is the way they look.' тройка (look, way, +2) описывает зависимость 'way' от 'look' и при этом показывает смещение одного слова относительно другого в этом тексте. Такие тройки вместе с частотами их встречаемости в исходном тексте формируют, так называемый, банк зависимостей.

Затем на твитах, состоящих только из словарных слов, обучается SVM-классификатор. При этом каждое слово представляется словами из своего ближайшего контекста и частотами в банке зависимостей троек вида (рассматриваемое слово, слово из контекста, расстояние между ними). Т.е. для предложения $...w_{i-3}, w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}, w_{i+3}, ...$ для слова w_i в качестве параметров используются слова $w_{i-3}, w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}, w_{i+3}$, а также частоты из банка зависимостей для троек (w_i, w_{i-j}, j), где $j = \overline{-3, 3}, j \neq 0$. Таким образом подготавливаются экземпляры, для которых классификатор должен принимать положительное решение. Экземпляры, у которых главное слово меняется на наиболее частотное слово из близких по расстоянию Дамерау-Левенштейна к нему, дополняют выборку отрицательными примерами.

Наконец, для определения является ли слово в исходном предложении некорректным, для каждого такого слова рассматриваются все слова из соответствующих множеств кандидатов. Если для слова-кандидата с контекстом, полученным из исходного предложения, классификатор выдает положительный ответ, то слово считается некорректным. Если же для всех кандидатов ответ оказывается отрицательным, то слово признаётся корректным и в дальнейшем не рассматривается.

Выбор оптимальной замены. Окончательный выбор наиболее подходящей замены происходит на основании множества различных характеристик, таких как фонетическое и лексическое расстояние, наиболее длинная общая последовательность букв, а также на основании языковой модели.

Тестирование данного метода происходило как на Твитах, так и на SMS-сообщениях. Авторам статьи удалось достичь полноты и точности метода порядка 0.75.

4.1.2 Модели ошибок

Ещё одним распространенным методом исправления ошибок является построение моделей ошибок, он используется, например, в статьях Брилла и Мур[8], статье Тоутанова и Мур[9] и статье Дутта, Саа и др.[14]. В этом случае задача представляется в терминах зашумлённого канала. Т.е. считается, что корректным текстом, который должен быть получен, в результате исправления, является исходное сообщение, но при передаче по каналу с шумом в сообщении возникают ошибки. Таким образом задача сводится к построению модели ошибок и модели языка.

В работе [8] используется модель для замен длиной до 5 символов. В отличие от своих предшественников, модель позволяет исправлять некоторые фонетические ошибки благодаря расширенному контексту, но при этом неправильно корректирует ошибки, связанные с произношением. Так, например, слово 'edelvise' исправляется на 'advice', вместо 'edelweiss', а слово 'latecks' на 'lacks' вместо 'latex'. В приведённых примерах искомое исправление произносится больше похоже на исходное слово, нежели на замену, предлагаемую этой моделью. Поэтому в статье [9] предлагается улучшение этого метода, заключающееся в построении дополнительной модели для фонетических ошибок.

Вернёмся к модели зашумлённого канала. Задача исправления ошибок заключается в сопоставлении каждому некорректному слову w корректного слова $r \in D$, где D — это словарь корректных слов, а r — наиболее вероятное слово из тех, что могут быть заменены при движении по каналу на w. Таким образом задача — найти r, такое что

$$r = \arg\max_{r' \in D} P(r'|w) = \arg\max_{r' \in D} \frac{P(r') * P(w|r')}{P(w)} = \arg\max_{r' \in D}(P(r') * P(w|r'))$$

В терминологии канала с шумом, P(r) относится к модели источника (в нашем случае это язык), а P(w|r) к модели канала (в нашем случае, модели ошибок).

В работе Брилла и Мура используется модель с параметрами $P(\alpha \to \beta|PSN)$, где α и β — исходная и результирующая последовательности букв длины не больше пяти, а PSN показывает положение исходной последовательности в слове. Исходное слово делится на части, длина каждой из которых не превосходит пяти. Каждая часть заменяется отдельно, а соответствующие вероятности перемножаются для получение вероятности изменения всего слова. Например, для слова 'bouncy', разбитого на две части 'boun cy', будут рассчитаны вероятности вида $P(boun \to boun) * P(cy \to cie)$. А вероятность P(w|r) оценивается как максимум вероятностей по всем разбиениям r, которые приводят к слову w.

Тоутанова и Мур предложили использовать дополнительно к такой модели ещё модель, обученную на транскрипциях слов. Для построения такой модели необходимо предварительно реализовать механизм, позволяющий для произволного слова построить его транскрипцию. В этой статье в качестве такого механизма используется обученная на исходных данных модель сопоставления символу последовательности звуков. Для этого используется набор слов с их транскрипциями, где каждому символу поставлена в соответствие часть транскрипции, а в качестве параметров используются соседние символы.

Тестирование в этих статьях происходило на 10 000 пар слов с ошибками и соответствующих корректных слов. Качество коррекции в обоих методах примерно одинаково и равно 94.22% и 95.58% соответственно.

4.1.3 Машинный перевод

Тот факт, что задача исправления ошибок в текстах и задача машинного перевода могут быть представлены в терминах канала с шумом, породил идею решать первую из них в терминах второй. Т.е. считать что исправление ошибок — это всего лишь перевод с языка с ошибками на более грамотный язык. В работе Кауфманна[2] для нормализации твитов применяется метод, основанный на использовании инструмента машинного перевода.

Метод работает в два этапа. Сначала происходит предобработка твитов, заключающаяся в орфографической нормализации и устранении синтаксической неоднозначности. После этого твиты подаются на вход машине автоматического перевода.

Тестирование и обучение проводится на коллекции твитов, вручную исправленных

десятью людьми. В процессе исправления из твитов удалены все элементы, не являющиеся необходимыми для формирования корректного предложения на английском языке. Такими элементами, например, являются смайлы и хештеги.

Предобработка заключается в исправлении некоторых наиболее распространённых типов ошибок. Сначала, при помощи таблицы сокращений, исправляются некоторые типичные для твитов акронимы. Затем осуществляется попытка исправить элементарные замены, вставки, удаления и перестановки символов. Также происходит исправление чрезмерных повторений одного и того же символа. Наконец определяется, является ли в данном случае использование тега или имени пользователя значимым в данном предложении. Если тег не является значимым, если он, например, просто перечислен среди других тегов в начале сообщения, он удаляется. В противном случае, например если тег используется в предложении в качестве обыкновенного слова, его необходимо оставить. Аналогично поступают и с упоминаниями пользователей.

На втором этапе происходит перевод с помощью средства автоматического перевода 'Moses', предварительно обученного на части твитов.

4.2 Исправление ошибок на символьном уровне

Методы исправления ошибок, основанные на анализе символьных последовательностей, более применимы в задаче распознавания текста[4]. Эта задача весьма близка к исправлению опечаток, но, при этом, имеет свои особенности. В случае распознавания ошибки, как правило, заключаются в неправильной интерпретации исходных символов. Опечатки же, тем более ошибки и намеренные искажения слов, на символьном уровне исправлять сложнее. Тем не менее подобные методы применяются для исправления ошибок. Рассмотрим наиболее свежие примеры их использования.

Метод исправления некоторых типов ошибок на символьном уровне

Метод, использованный в статье Эскандера [11] заключается в подготовке отдельных классификаторов для наиболее распространённых ошибок и их каскадном применении. Данный метод обладает существенным недостатком, который заключается в сильной зависимости метода от конкретного языка. При попытке адаптировать метод к языку отличному от арабского придётся заново формировать набор классификаторов на

основании информации об ошибках типичных для данного языка, т.е. по сути разрабатывать метод заново. В статье Фарра [3] приведён метод, весьма похожий на данный, но при этом использующий более общие предположения, не зависящие от конкретного языка.

Обобщённый метод исправления ошибок на символьном уровне

В отличие от предыдущего данный метод использует одну модель для всех типов ошибок. Этот метод является контекстночуствительным, т.е. при исправлении ошибки учитываются слова, соседние с рассматриваемым. Метод, в представленном в статье[3] виде, использует только независимые от конкретного языка параметры для построения классификатора, но может быть легко модифицирован ориентированными на целевой язык параметрами.

Общая схема работы метода. Метод заключается в сведении задачи исправления ошибок к задаче маркировки всех букв специальными метками, описывающими, что нужно сделать с этим символом для получения корректного его аналога. Используются метки следующих видов:

- OK — не нужно предпринимать никаких действий,
- ЗАМЕНИТЬ_НА(c) — данный символ следует заменить на символ "c",
- УДАЛИТЬ — символ нужно удалить,
- ВСТАВИТЬ(c) — перед этим символом следует вставить символ "c".

Метки расставляются SVM-классификатором. При этом пробел считается обычным символом и тоже помечается.

В таблице 1 приведен пример расстановки пометок для слова "korrectd". В данном примере первый символ слова заменяется на "c", перед буквой "r" вставляется ещё одна, а последний символ удаляется. Все остальные символы остаются без изменений, и в итоге получается слово "correct".

Таблица 1: Расстановка пометок на примере слова ”korectd”->”correct”

Слово	**Метки**
k	ЗАМЕНИТЬ_НА(c)
o	ОК
r	ВСТАВИТЬ(r)
e	ОК
c	ОК
t	ОК
d	УДАЛИТЬ

Параметры классификатора. В качестве базовых параметров используются исходный символ, два предшествующих и два последующих символа, а также два первых и два последних символа слова.

Метод максимального правдоподобия. В качестве дополнения к основному алгоритму используется метод максимального правдоподобия, использующий юниграмную модель языка и пытающийся заменить каждое слово на наиболее вероятный из корректных его аналогов.

Тестирование методов исправления ошибок на символьном уровне

Описание данных. Модель тестировалась на текстах на египетском диалекте арабского языка. В этом корпусе около 81% ошибок — это замены одного символа на другой, ещё примерно 15% — вставка и удаление одного символа. Оставшиеся 4% ошибок являются более сложными, и рассматриваемым методом не исправляются. Обучение классификатора происходит на параллельных текстах. В исходных и исправленных текстах предварительно установлена связь между соответствующими символами.

Оценка моделей. Оценка моделей производилась по метрике корректности, которая представляет из себя отношение количества правильных слов в полученном в результате работы программы тексте к общему количеству слов. В качестве базового решения использовалась программа, не изменяющая исходный текст. Результаты тестирования методов Фарра[3] и Эскандера[11] приведены в таблице 2. Для более по-

дробного анализа качества работы методов приведены результаты их работы на словах из обучающей выборки и на новых для системы словах.

Таблица 2: Результаты тестирования методов исправления ошибок на символьном уровне.

Подход	Аккуратность, %	Известные, %	Новые, %
Базовое решение	77.2	78.6	70.7
Метод Эскандера[11]	91.5	94.6	76.5
Метод Фарра[3]	92.2	94.6	80.5

4.3 Выводы

В данном разделе были рассмотрены существующие методы исправления ошибок в текстах на естественных языках. Рассмотренные методы разделены на две группы: исправляющее ошибки на символьном уровне и на уровне слов. Первая группа методов использует информацию о соседях символа и о его положении в слове для того, чтобы принять решение о его исправлении. Вторая группа использует словарь корректных слов. Методы исправления ошибок на символьном уровне способны исправлять ошибки в неизвестных словах, но нередко осуществляют лишние исправления. Методы второй категории, напротив, гарантируют, что слова из словаря исправляться не будут, а с исправлением новых слов такими методами часто оказывается невозможным.

Методы, основанные на модели зашумлённого канала, в том числе методы, применяющие средства машинного перевода, сильно зависят от обучающих данных. Кроме того, для обучения они требуют довольно большого количества данных. Методы основанные на генерации множеств кандидатов более предсказуемы, и для работы требуют, по большому счёту, только словарь корректных слов. Методы этого типа более структурированы и, соответственно, в них проще встраивать дополнительные компоненты, что позволяет сделать предположение о возможности увеличения качества работы данного метода посредством применения в нём некоторых идей из других исследований.

5 Исследование и построение решения задачи

5.1 Базовый метод

В качестве базового используется метод, реализованный в системе анализа текстов Texterra[18], разрабатываемой Институтом Системного программирования РАН. Этот метод, реализованный для английского языка, адаптирован под русский язык заменой всех моделей без изменений в схеме работы системы. Метод состоит из трёх этапов: обнаружение ошибок, генерация множества кандидатов, выбор оптимальной замены для каждой ошибки.

Обнаружение ошибок происходит с помощью словаря. Все слова, не содержащиеся в словаре, считаются ошибками. Затем происходит генерация множеств кандидатов с помощью трёх различных методов:

- Из словаря выбираются слова такие, что расстояние Дамерау-Левенштейна от каждого из них до исходного слова не превосходит единицы.
- Для всех повторений одной буквы подряд более двух раз генерируются два кандидата, в одном эта буква повторяется дважды, в другом не повторяется вовсе.
- Фонетические замены происходят на основании фонетического словаря. Каждая последовательность букв в словаре заменяется на соответствующий номер класса, и сравниваются полученные фонетически шаблоны. Например, 'счётчик' и 'щётчик' будут иметь один и тот же шаблон '<i>ётчик', где '<i>' метка класса, к которому принадлежат последовательности 'сч' и 'щ'. Следовательно, в данном случае, слова 'счётчик' и 'щётчик' будут считаться одинаковыми, и второе будет заменено на первое если встретится в тексте.

На заключительном этапе происходит выбор одного слова из каждого множества кандидатов. Для этого для каждого из неправильных слов производится процедура выбора наиболее подходящего по контексту слова на основании 3-граммной модели языка.

5.2 Общая схема реализованного метода

Метод, реализованный в данной работе, имеет четрёхэтапную схему работы (см. 1).

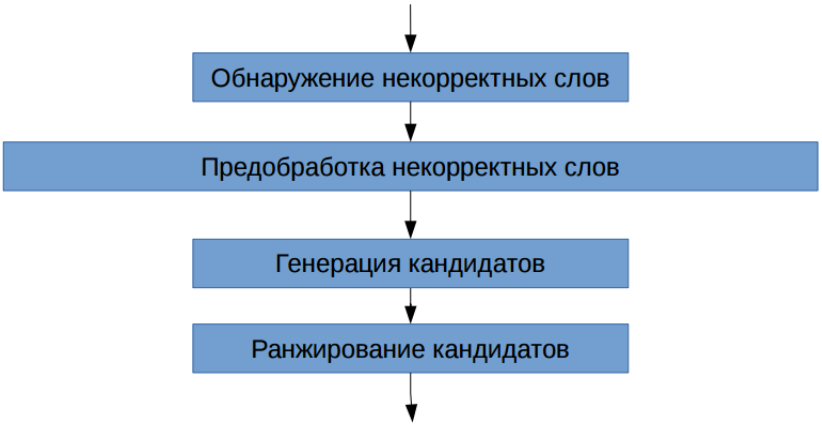

Рис. 1: Общая схема разработанного метода.

Сначала в исходных предложения выделяются слова, в которых, вероятно, есть ошибки. После этого каждое некорректное слово проходит предобработку для дополнительной разметки. Затем для каждого из этих слов генерируется множество вероятных замен. Это множество состоит из корректных слов, по некоторым признакам похожих на данное некорректное. На заключительной стадии, на основании стоимости каждого из сгенерированных вариантов, а также соответствия замены исходному контексту, выбирается наиболее вероятная замена. Если для некорректного слова не получилось сгенерировать ни одного корректного кандидата, то считается что исходное слово корректно, но отсутствует в нашем словаре (например, является неологизмом или окказионализмом) и, соответственно, замены не происходит. Рассмотрим подробнее каждый из этих четырёх этапов.

5.3 Выявление ошибок

Выявление ошибок происходит на основе словарей. В качестве основы для словаря используется словарь словоформ, сгенерированный с помощью "Грамматического словаря русского языка А. А. Зализняка". Для пополнения этого словаря корректными разговорными словами используется тренировочный корпус. Так, все слова, встречающиеся в корректной версии предложений, из тестовой выборки включены в словарь.

Все слова в предложении просматриваются независимо друг от друга, и каждое из них проверяется на присутствие в словаре. Если слово встретилось в словаре, то оно заведомо корректно, хотя может и не подходить по контексту в данном предложении.

Если же слово не встретилось в словаре, то либо оно некорректно, либо редко или не так давно встречается в языке и, соответственно, надо попробовать его исправить. Таким образом в этой работе исправляются только ошибки, приведшие к образованию хотя бы одного некорректного слова. А ошибки вида 'кот'->'кто' и, например, 'что то'->'что-то' не исправляются, так как применяемый метод не позволяет их обнаружить.

5.4 Предобработка некорректных слов

На этом этапе используется идея, изложенная в статьях Фарра[3] и Эскандера[11]. Она заключается в рассмотрении предложения как последовательности символов и снабжении каждого элемента этой последовательности одной из следующих меток (всего 70 штук):

- ВСТАВИТЬ(c) — обозначает, что для исправления ошибки необходимо вставить символ 'c' перед помеченным (всего 32 метки для символов русского алфавита и две для дефиса и пробела, итого 34),
- ЗАМЕНИТЬ_НА(c) — необходимо заменить помеченный символ на 'c' (ещё 34 метки)
- УДАЛИТЬ — удалить помеченный символ (1 метка)
- ОК — этот символ следует оставить без изменений (1 метка)

В реализованном методе все некорректные слова проходят процедуру предобработки, в процессе которой каждая буква получает одну из этих меток. Рассмотрим пример работы этой процедуры в случае исправления слова 'каректнно' на 'корректно' (см. таблицу 3).

Таблица 3: Расстановка пометок на примере слова 'каректнно'->'корректно'

Слово	**Метки**
к	OK
а	ЗАМЕНИТЬ_НА('о')
р	ВСТАВИТЬ('р')
е	OK
к	OK
т	OK
н	OK
н	УДАЛИТЬ
о	OK

Для решения этой задачи обучается классификатор. В качестве меток классов используются описанные выше 70 различных помет. Параметрами классификатора служат три предшествующих и три последующих символа для помечаемого, также два первых и два последних символа в текущем слове. Границы слов при выборе предшествующих и последующих символов для данного не учитываются, т.е. если, например, рассматривается первая буква слова, то предшествующими символами считаются пробел и последние буквы предыдущего слова. Классификатор обучается на ошибках их тренировочного корпуса. При этом ошибки, которые нельзя описать с помощью таких пометок, например пропуск нескольких букв, при обучении не учитываются.

Данный классификатор используется для разметки букв в некорректных словах. При этом используется не только само слово, но и ближайший его контекст. Полученные метки в дальнейшем будут использоваться при подсчёте модифицированного расстояния Левенштейна для дисконтирования стоимости преобразований, предсказанных этой моделью.

5.5 Генерация кандидатов для замены

На этом этапе происходит генерация слов, которые могли быть корректным аналогом слова с ошибкой. Сначала на основании некоторого метода генерируются новые

слова, а затем полученные слова проверяются на присутствие в словаре. Все словарные слова, которые удалось сгенерировать для некоторого некорректного слова, образуют множество кандидатов для замены этого слова. По окончании данного этапа, каждый кандидат также имеет численную характеристику — стоимость преобразования из исходного слова. В данной работе для генерации кандидатов используются три различных метода. Каждый из методов генерирует своё множество кандидатов, после чего эти множества сливаются в одно. При этом если один кандидат присутствует в нескольких множествах, то он добавляется в результирующее множество единожды, а в качестве его стоимости выбирается наименьшая из его стоимостей в объединяемых множествах.

5.5.1 Ближайшие по расстоянию Дамерау-Левенштейна

Классическое расстояние Дамерау-Левенштейна.
Расстояние Дамерау-Левенштейна является одним из самых старых, но при этом и одним из наиболее часто применяемых орудий для исправление опечаток. Ещё в середине 60-х годов прошлого века Дамерау [5] заметил, что более 80% ошибок в текстах представляют собой ошибку одного из следующих четырёх типов: перестановку двух соседних символов, замену одного символа на другой, вставку или удаление одного символа. Примерно в это же время была опубликована работа Левенштейна [6], в которой он определил метод вычисления расстояния между символьными последовательностями, позднее названный его именем. Хотя результаты Дамерау и были получены для текстов на английском языке, тем не менее, "расстояние Дамерау-Левенштейна с некоторыми модификациями или даже без них применимо к множеству различных языков, включая персидский, арабский, испанский, баскский и другие"[7].

В 2014 году Гименес, Роман и Карвальо [7] провели исследование типичных ошибок для бразильского диалекта португальского языка. В результате оказалось, что 76,2% ошибок в корпусе — это одна вставка, удаление, перестановка символа или диакритического знака. А более 91% ошибок — это не более, чем двойная ошибка одного из этих типов, либо вставка, удаление или перестановка пробела.

Расстояние Дамерау-Левенштейна в самом простом случае представляет собой минимальное количество преобразований четырёх базовых типов, которое преобразует одно из слов в другое. Например для слов 'хочеца' и 'хочется' расстояние Дамерау-Левенштейна равно трём: можно, например, вставить букву 'т' в первое слово, затем

заменить 'ц' на 'с', а 'а' на 'я'.

Модификация расстояния Дамерау-Левенштейна. В данной работе рассматривается модифицированное расстояние Дамерау-Левенштейна. Как уже говорилось в разделе 5.4, метки, полученные в процессе предобработки некорректных слов, используются для дисконтирования некоторых преобразований на данном этапе. Если символ имеет метку 'УДАЛИТЬ', то его удаление стоит не единицу как в общем случае, а несколько меньше — 0.65. Аналогично дело обстоит с вставками и заменами: если модель их предсказала — значит они дешевле. Такое значение выбрано для того, чтобы три преобразования, предсказанные моделью, стоили чуть дешевле, чем два непредсказанных, но при этом два предсказанных всё же дороже одного обычного. Такая модификация, как будет сказано в разделе 5.9, даёт небольшой прирост качества.

Кроме того, в данной работе рассматривается обобщённый вариант расстояния Дамерау-Левенштейна, т.е. считается расстояние не для слов, а для слов с контекстом, что позволяет исправлять ошибки в расстановке пробелов и дефисов.

Рассчёт модифицированного расстояния Дамерау-Левенштейна. Генерация кандидатов происходит рекурсивно. На каждой итерации генерируются все слова, которые можно получить одним преобразованием из данного. Для каждого из таких слов считается его стоимость как сумма стоимости данного и стоимости преобразования (равна 1 или 0.65). Если стоимость получается больше заданной (в данной работе оно выбрано равным 2, так как большая часть некорректных слов находится на расстоянии 1-2 от своего корректного аналога, см. раздел 5.8), то полученное слово не рассматривается. Затем, если слово содержится в словаре (вернее говоря, каждое из полученной последовательности слов содержится в словаре, ведь мы можем вставлять пробелы), то оно добавляется в список вероятных замен. После чего рекурсивно запускается процедура генерации для полученного слова. В результате работы алгоритма мы получим список словарных слов находящихся на расстоянии не более двух от исходного.

Например, для слова 'ктота', с пометками {OK, OK, OK, ВСТАВИТЬ(' '), ЗАМЕНИТЬ_НА('о')} может быть сгенерировано множество замен {'кто-то': 1.65, 'кто то': 1.3, 'квота': 1, 'крот': 2}. В первом случае замена 'а' на 'о' предсказана символьной моделью и, следовательно, дисконтируется, вставка же дефиса не предсказана и стоит единицу, значит суммарная стоимость $1 + 0.65 = 1.65$. Во втором случае, оба преобразования предсказаны моделью, и значит стоимость равна $0.65 + 0.65 = 1$. Случаи 'квота'

и 'крот' не отвечают предсказаниям модели и, следовательно, дисконтирование их не касается.

5.5.2 Устранение излишних повторений букв и слогов

Весьма часто в сообщениях из социальных сетей встречается чрезмерное повторение одной и той же буквы, например 'о' в 'дооолгооо' или 'оочень' и 'а' в 'каааааак'. Несколько реже происходит случайный повтор одного слога, как правило два раза, например 'че' в 'количечеству'. При этом корректный аналог такого слова может находится весьма далеко по расстоянию Дамерау-Левенштейна от исходного слова. Так, например, для приведённого выше слова 'дооолгооо', корректный аналог находится на расстоянии 4, хотя с точки зрения человека искажений в этом слове всего два. А в случае слова 'количечеству' расстояние Дамерау-Левенштейна равно двум, хотя ошибка в слове всего одна. Данный генератор кандидатов нацелен как раз на исправление такого рода ошибок и позволяет уменьшить стоимость соответствующих кандидатов.

Если в некорректном слове встречается три или более одинаковых буквы подряд, то скорее всего ошибка кроется именно в чрезмерном повторении этой буквы. Поэтому в таких случаях мы генерируем двух кандидатов с одной и двумя буквами соответственно. Причём, так надо поступить с каждым чрезмерным повторением некоторой буквы в слове и, более того, стоит рассматривать и вариант повторения буквы всего два раза. Например для слова 'дооолгооо' будут сгенерированы варианты 'долго', 'долгоо', 'доолго' и 'доолгоо', и все кроме первого отсеются при проверке на вхождение в словарь. А для слова 'оочень' будут получены варианты 'оочень' и 'очень', причём словарным является только второй. Заметим, что в результате для слов 'дооолгооо' и 'оочень' стоимость замены будет 2 и 1 соответственно.

Ситуация со слогами обстоит чуть сложнее. В рамках данной работы будем считать слогом не совсем то, что подразумевается под этим термином в русском языке. Определим слог, как гласную, к которой могут быть приписаны (как слева, так и справа) согласные. Обозначим его как xay, где x и y — последовательности согласных букв, а a — гласная буква. Тогда повторение слога представимо в виде $xayxay$. Таким образом, наша задача заключается в том, чтобы перебрать все пары одинаковых гласных, которые в нашем слове разделены только согласными. Для каждой такой пары необходимо перебрать все возможные варианты поделить согласные между ними на начало и конец

слога. И, наконец, необходимо проверить, что начала и концы слогов совпали. Вернёмся к примеру 'количечеству'. Не сложно заметить, что в нашем примере только в одном случае две соседние гласные одинаковы и, следовательно, есть два варианта выбрать слог для этих гласных: 'е|че' и х='ч', у=", слог 'че' и 'еч|е' и х=", у='ч', слог 'еч'. Теперь необходимо проверить, повторяются ли в слове именно эти слоги, т.е. проверить, содержит ли исходное слово последовательность букв 'чече' или 'ечеч'. В нашем случае только первая последовательность содержится в слове, и она же приводит нас к корректной замене. При этом стоимость замены будет равна единице.

5.5.3 Простые фонетические замены

В целях получения более короткого написания, по неграмотности или по иным причинам, слова в тексте могут приобретать искажённое написание. Как правило, новое написание очень близко или полностью совпадает с исходным по произношению. Идея данного метода в том, чтобы заменять последовательности букв в исходном слове на последовательности букв схожих по звучанию. Для этой цели был создан словарь схожих по звучанию последовательностей букв на основании материалов [16] и [17]. В таблице 4 приведены все эквивалентные последовательности букв, а также примеры слов соответствующие этим последовательностям.

Таблица 4: Основа фонетического словаря

Эквивалентные последовательности	**Примеры**
сч, ш, щ, зч, здч	счётчик, грузчик, объездчик
тс, ц, дс, тьс	купаться
гк, х	легкий
тщ, чш	отщепнуть
сц, здц	под уздцы
стс, сс, с	бестселлер, постскриптум, журналистский

Также в словарь в процессе обучения добавляются все сложные замены из тренировочного корпуса, например пары 'лутчие'-'лучшие', 'раене'-'районе' породят эквивалентные последовательности 'тч'-'чш' и 'е'-'йо'. Исходя из полученного словаря экви-

валентностей, происходит подбор слов скорее всего схожих по звучанию с данным и присутствующих в словаре.

5.6 Выбор оптимальной замены

Выбор оптимальной замены происходит в два этапа (см. рисунок 2). Сначала отсеиваются слишком дорогие, по сравнению с остальными, кандидаты. А затем среди оставшихся кандидатов выбирается наиболее подходящий по контексту.

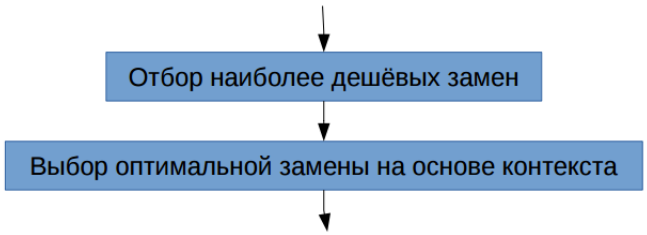

Рис. 2: Схема выбора оптимальной замены.

5.6.1 Отбор слов с наименьшей стоимостью преобразования.

На данном этапе для каждого некорректного слова мы имеем множество слов-кандидатов со стоимостями их получения. Стоимости — это вещественные числа большие 0 и не превосходящие 2. Соответственно, нам надо выбрать слова достаточно близкие по стоимости к наиболее дешёвому варианту. Для этого мы сортируем слова в каждом множестве в порядке возрастания их стоимостей и отбираем только те, которые отличаются от первого по стоимости не больше, нежели на 0.5. Для примера из 5.5.1 ({'кто-то': 1.65, 'кто то': 1.3, 'квота': 1, 'крот': 2}) после этого шага множеством кандидатов будет {'кто то': 1.3, 'квота': 1}

5.6.2 Выбор оптимальной замены на основе n-граммной модели языка

Для окончательного выбора наиболее подходящего слова используется 3-граммная модель языка, обученная на корректных предложениях из тренировочного корпуса. Для сглаживания используются n-граммные модели меньшего порядка. Так как в предложении может быть более одной ошибки, то в контекст для некорректного слова может по-

пасть другое слово с ошибкой. В таком случае нам придётся перебирать все возможные комбинации кандидатов для этих некорректных слов. В общем случае получаем экспоненциальную сложность по времени. Поэтому для особенно сложных предложений (с произведением длин множеств кандидатов больше 100) выбор кандидатов оптимальных замен происходит последовательно. Т.е. сначала выбирается слово, для которого проще всего выбрать замену. Для этого сначала для каждого из некорректных слов считаются следующие величины: разности между стоимостями двух наиболее дешёвых замен для каждого из слов (1), длины списков замены (2), отношения частот двух наиболее распространённых слов в каждом множестве (3) и средние длины слов в каждом множестве (4). После этого слова сортируются по этой четвёрке величин: по убыванию (1), при равенстве (1) — по возрастанию (2), затем по убыванию (3) и по убыванию (4). В этом списке слова отсортированы по сложности выбора оптимального кандидата. Дальнейший выбор происходит последовательно. При этом для очередного слова используется исходное предложение, в котором произведены все оптимальные замены для более простых (в смысле выбора оптимальной замены) слов.

5.7 Полная схема работы реализованного метода

На рисунке 3 приведена полная схема реализованного метода.

5.8 Описание данных для обучения и тестирования

Для обучения и тестирования метода использовалась „выборка предложений из текстов в социальных сетях“, предоставленная ресурсом Web Corpora [1]. Обучающий и тестовый корпуса содержат по 2000 и 2007 предложений соответственно. Для каждого предложения имеется его корректный аналог, исправленный экспертами. Как показал анализ, примерно 40% предложений совсем не содержат ошибок. Около 40% предложений тренировочного корпуса и более 30 тестового содержат ровно по одной ошибке. А предложений, содержащих больше двух ошибок, вообще меньше 10%. Распределение предложений по количеству ошибок приведено в таблице 5.

[1] http://www.webcorpora.ru/news/215

Рис. 3: Общая схема разработанного метода.

Таблица 5: Распределение предложений по количеству содержащихся в них ошибок

Количество ошибок	0	1	2	3	4	5	6	7
Тренировочный корпус	818	803	279	79	14	12	3	0
Тестовый корпус	804	666	379	106	38	11	3	1

В данных корпусах, 80% ошибок оказались заменами одного слова на другое. Более 18% ошибок это замена одного слова на два (устранение ”склеивания” двух слов) и двух слов на одно (удаление лишнего пробела). На более сложные замены приходится менее двух процентов ошибок. Подробная информация представлена в таблицах 6 и 7. В столбцах - количество слов в замене, в строках — количество слов в исходном варианте.

Таблица 6: Распределение ошибок по количеству слов в исходном и исправленном вариантах для тестового корпуса.

	0	1	2	3
0	0.0000	0.0507	0.0507	0.0000
1	0.1013	79.4326	11.6008	0.1013
2	0.0507	8.2067	0.2026	0.0000
3	0.0000	0.2026	0.0000	0.0000

Таблица 7: Распределение ошибок по количеству слов в исходном и исправленном вариантах для тренировочного корпуса.

	0	1	2	3
0	0.0000	0.0577	0.0000	0.0000
1	0.1155	79.9654	15.1270	0.4619
2	0.0000	3.9838	0.1732	0.0000
3	0.0000	0.0577	0.0000	0.0000
4	0.0000	0.0000	0.0000	0.0000
5	0.0000	0.0577	0.0000	0.0000

Вставка и удаление слов (первая строка и первый столбец в таблицах) в данном корпусе представляют собой вставку или пропуск предлогов.

Среди наиболее сложных ошибок (в смысле количества слов в исходном и в исправленном вариантах) стоит выделить неправильную расстановку дефисов и пробелов в сложных словах (точь в точь/точь-в-точь, вобщем то/в общем-то, дооолго долго долго/долго-долго-долго-долгоб, не на долга/ненадолго и подобные) и ошибки, вызванные перестановкой пробела и соседнего с ним символа, в результате чего буква от одного слова приклеивается к другому (пос толу/по столу, ка кв/как в)

В таблице 8 приведено распределение ошибок по расстоянию Дамерау-Левенштейна между исходным словом и его исправлением. Заметим, что более 90% исправлений

лежат на расстояние не более двух от исходного слова. Наиболее сложные исправления представляют из себя чрезмерные повторения букв (ооооочень/очень, покажиииии/покажию, ээххх/эх, мааааммоооочкиииии/мамочки) и намеренные искажения слов (собсна/собственно, ессно/естественно)

Таблица 8: Распределение ошибок по расстоянию Дамерау-Левенштейна между исходным словом и его исправлением

Расстояние	1	2	3	4	5	6	7	8	9	10
Тренировочный корпус	1395	211	78	32	11	4	0	0	0	1
Тестовый корпус	1671	205	64	19	8	0	1	0	1	1

5.9 Способ и результаты тестирования

Для оценки результатов работы представленных методов использовались стандартные метрики:

- точность — отношение количества корректно исправленных ошибок к общему количеству ошибок
- полнота — отношение количества корректно исправленных ошибок к общему количеству исправлений
- стандартная F1-мера, $F_1 = \frac{2*p*r}{p+r}$, где, p — точность, r — полнота.

Результаты тестирования приведены в таблице 9. Первая строка — результаты тестирования базового метода, описанного в разделе 5.1. Каждая из последующих строк представляет собой результаты для улучшения метода из предыдущей.

Автоматическое пополнение словарей на основании тренировочного корпуса значительно улучшило как полноту, так и точность метода. Многие разговорные слова в базовом варианте исправлялись на слова из словаря, что отрицательно сказывалось на полноте метода. А для некоторых слов в словарях не оказывалось варианта необходимого для исправления, в результате чего слово или не исправлялось вовсе, или исправлялось не на то слово.

Решение, помимо автоматического построения словарей, реализующее ещё исправление ошибок, вызванных неправильной расстановкой пробелов, также улучшило качество исправления ошибок по сравнению с предыдущей версией метода. Наконец, финальное решение, помимо прочего использующее модель символьных ошибок для предварительной обработки некорректных слов с целью последующего дисконтирования предсказанных преобразований при подсчёте расстояния Дамерау-Левенштейна, также улучшило качество метода.

Таблица 9: Результаты тестирования различных методов.

Решение	**Точность**	**Полнота**	**F1-мера**
Базовый метод	55,5	47,4	51,1
+ Автоматическое построение словарей	60,3	53,5	56,7
+ Исправление ошибок в расстановке пробелов	66,6	57,3	61,6
+ Дисконтирование предсказанных исправлений	69,5	59,4	64,1

6 Описание практической части

6.1 Обоснование выбранного инструментария

Программное средство, разработанное в рамках данной работы для исправления ошибок в предложениях, реализовано на языке Python, поскольку это высокоуровневый язык программирования, и для него существует множество открытых библиотек, упрощающих разработку программ.

Для построения n-граммных моделей языка использовалась библиотека анализа текстовых данных nltk. Классификатор SVM, использованный для разметки символов в процессе предобработки, взят из библиотеки scikit-learn.

6.2 Схема работы программной системы

На рисунках 4, 5 и 6 приведена схема работы реализованного программного средства для исправлении ошибок, для удобства восприятия разбитая на три части. Овалами серого цвета на схемах обозначены части алгоритма, белыми прямоугольниками — данные, используемые или генерируемые алгоритмом. Прямоугольники, соответствующие входным и выходным данным схемы, имеют пунктирную границу, а соответствующие заранее подготовленным данным (словарям и моделям) — границу из точек и штрихов. Остальные прямоугольники соответствуют промежуточным представлениям данных и имеют сплошную границу.

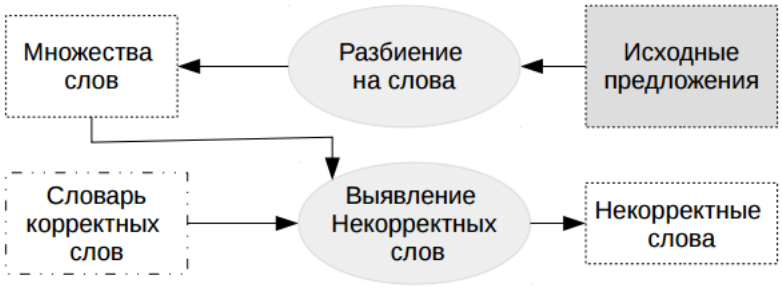

Рис. 4: Схема выявления некорректных слов.

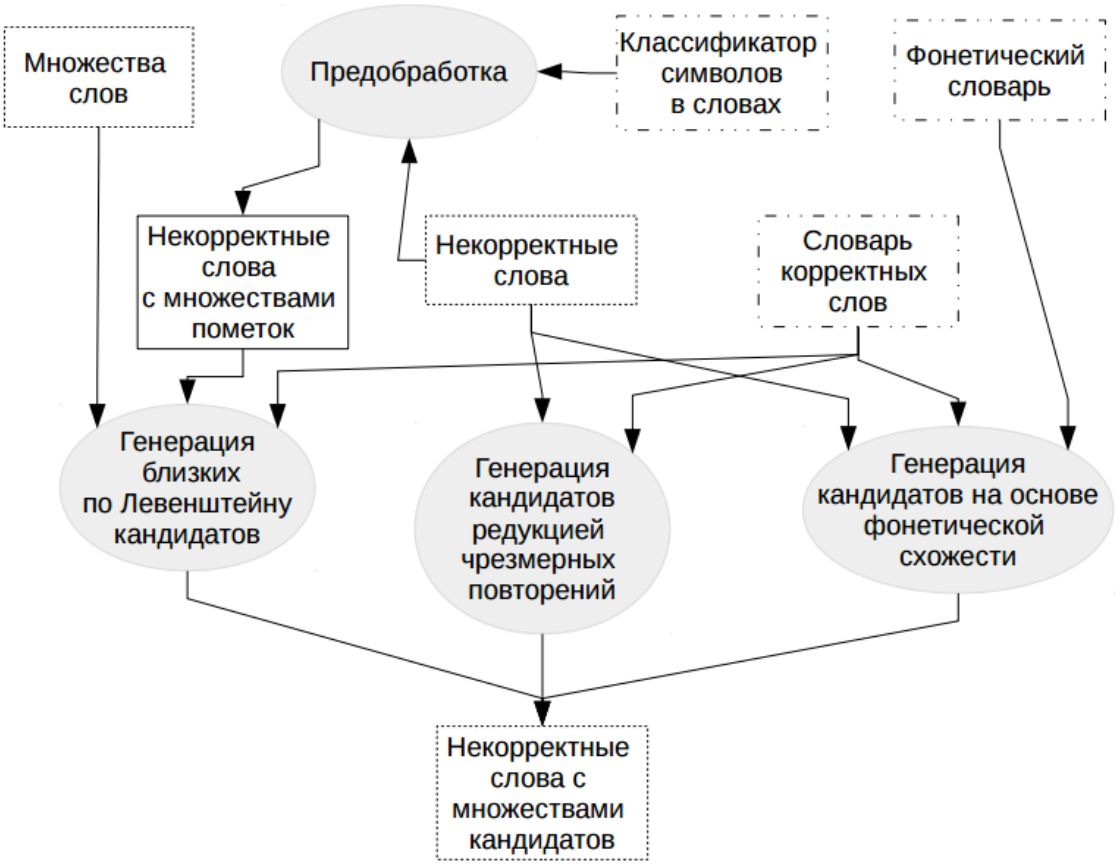

Рис. 5: Схема генерации множеств кандидатов.

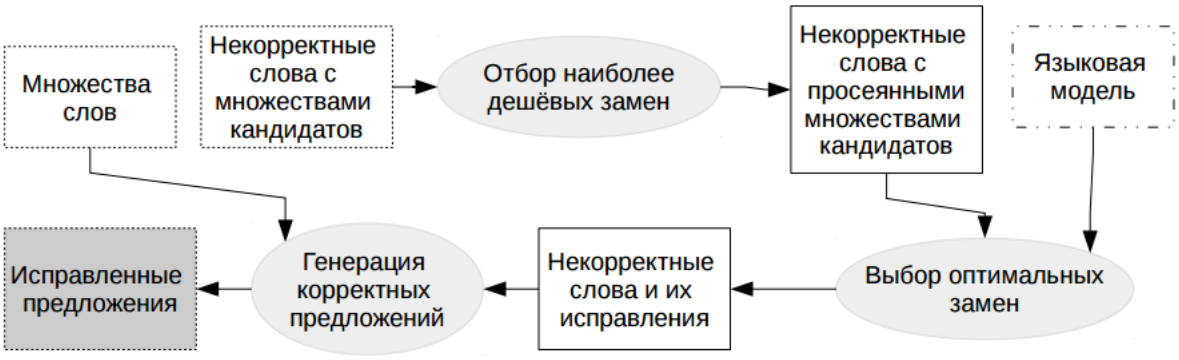

Рис. 6: Схема выбора оптимальной замены.

На рисунке 7 представлена общая схема работы программного средства.

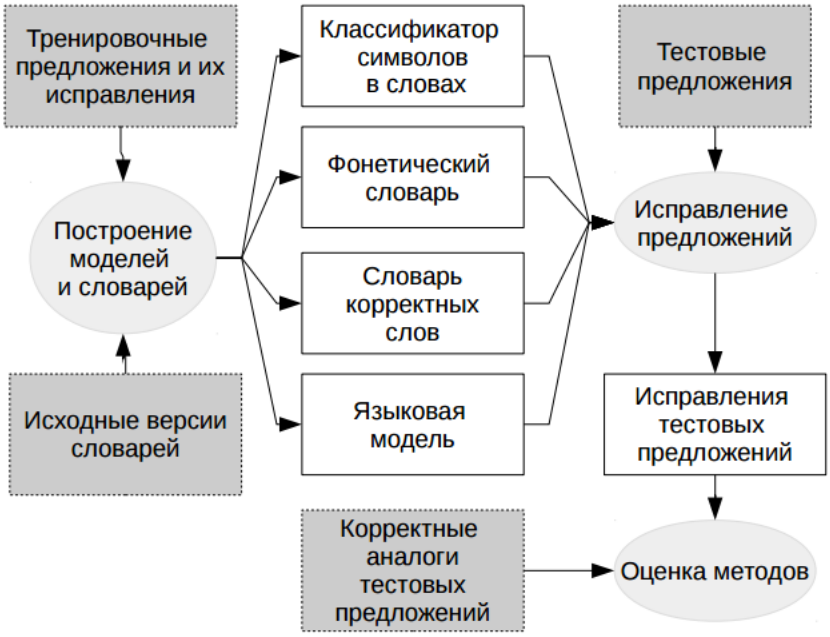

Рис. 7: Общая схема программной системы.

6.3 Общая архитектура

На рисунке 8 представлена общая схема модулей реализованного программного средства.

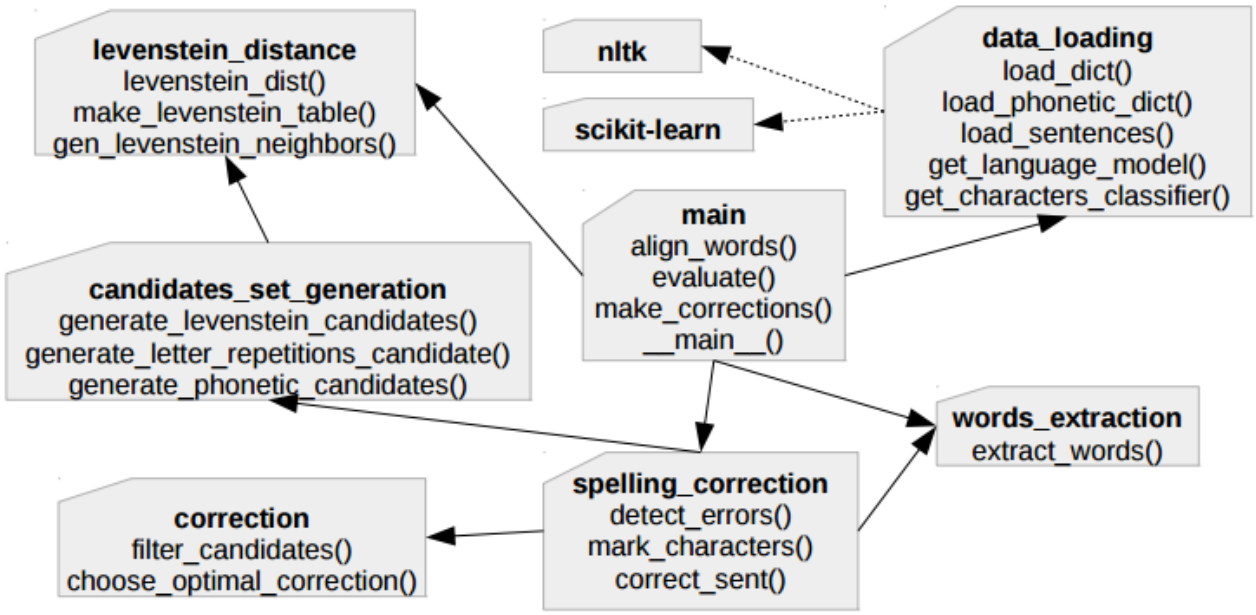

Рис. 8: Общая схема программной системы.

- main
  align_words() — устанавливает оптимальное соответствие между двумя предложениями. Для этого считает таблицу Левенштейна для множества слов.
  evaluate() — считает полноту, точность и F-меру на основании полученного в результате работы программы и составленного экспертами множеств ошибок.
  make_corrections() — осуществляет исправление ошибок для заданного множества предложений, а также извлекает исправления из золотого стандарта и некорректных предложений. __main__() — выполняет загрузку предложений, моделей и словарей. Затем запускает исправление ошибок в загруженных предложениях и производит оценку качества исправления.

- data_loading — занимается загрузкой данных
  load_dict() — загрузка обычного словаря из файла. И, в зависимости от входных

параметров, может также выполнять его пополнение.
load_phontetic_dict() — загрузка фонетического словаря из файла и, возможно, его пополнение
load_sentences() — загрузка предложений.
get_language_model() — загрузка языковой модели или её расчёт (в зависимости от параметров запуска).
get_characters_classifier() — загрузка и обучение классификатора символов.

- spelling_correction — основной модуль исправления ошибок
  detect_errors() — обнаружение ошибок в предложениях.
  mark_characters() — разметка символов с помощью классификатора.
  correct_sent() — исправление предложения.

- words_extraction
  extract_words() — преобразует предложение в множество слов.

- correction — модуль выбора оптимальной замены.
  filter_candidates() — осуществляет отсеивание лишних кандидатов.
  choose_optimal_correction() — выбирает оптимальную замену с помощью языковой модели.

- candidates_set_generation — расчёт множеств кандидатов.
  generate_levenstein_candidates() — варианты исправления на основании близости по Левенштейну
  generate_letter_repetitions_candidate() — устранение чрезмерных повторений.
  generate_phonetic_candidates() — исправление фонетических ошибок.

- levenstein_distance — расчёт расстояния Дамерау-Левенштейна.
  levenstein_dist() — считает расстояние между двумя словами или двумя множествами слов.
  make_levenstein_table() — рассчитывает таблицу Левенштейна.
  gen_levenstein_neighbors() — генерирует ближайших для данного слова соседей.

7 Заключение

В рамках выпускной квалификационной работы были получены следующие результаты:

- проанализированы современные методы обнаружения и исправления ошибок в текстах на естественных языках;
- разработан метод исправления ошибок в предложениях, осуществляющий проверку слов по словарю, подбор вариантов исправления некорректных слов и выбор оптимального варианта на основе n-граммной модели языка;
- спроектировано и построено программное средство, реализующее данный метод;
- проведён сравнительный анализ результатов работы реализованного средства и более простого решения данной задачи и выявлено значительное улучшение качества исправление ошибок в предложениях.

Список литературы

[1] *Han B., Baldwin T.* Lexical normalisation of short text messages: Makn sens a# twitter //Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. – Association for Computational Linguistics, 2011. – С. 368-378.

[2] *Kaufmann M., Kalita J.* Syntactic normalization of twitter messages //International conference on natural language processing, Kharagpur, India. – 2010.

[3] *Farra N. et al.* Generalized Character-Level Spelling Error Correction //ACL (2). – 2014. – С. 161-167.

[4] *Kukich K.* Techniques for automatically correcting words in text //ACM Computing Surveys (CSUR). – 1992. – Т. 24. – №. 4. – С. 377-439.

[5] *Damerau F. J.* A technique for computer detection and correction of spelling errors //Communications of the ACM. – 1964. – Т. 7. – №. 3. – С. 171-176.

[6] *Levenshtein V. I.* Binary codes capable of correcting deletions, insertions, and reversals //Soviet physics doklady. – 1966. – Т. 10. – №. 8. – С. 707-710.

[7] *Gimenes P. A., Roman N. T., Carvalho A. M. B. R.* Spelling error patterns in brazilian portuguese //Computational Linguistics. – 2015.

[8] *Brill E., Moore R. C.* An improved error model for noisy channel spelling correction //Proceedings of the 38th Annual Meeting on Association for Computational Linguistics. – Association for Computational Linguistics, 2000. – С. 286-293.

[9] *Toutanova K., Moore R. C.* Pronunciation modeling for improved spelling correction //Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. – Association for Computational Linguistics, 2002. – С. 144-151.

[10] *Большаков И. А.* Проблемы автоматической коррекции текстов на флективных языках //Итоги науки и техники. Серия «Теория вероятностей. Математическая статистика. Теоретическая кибернетика». – 1988. – Т. 28. – №. 0. – С. 111-139.

[11] *Eskander R. et al.* Processing Spontaneous Orthography //HLT-NAACL. – 2013. – С. 585-595.

[12] *Гниловская Л. П., Гниловская Н. Ф.* Автоматическая коррекция орфографических ошибок. – 2004.

[13] *Chaudhuri B. B.* Reversed word dictionary and phonetically similar word grouping based spell-checker to Bangla text //Proc. LESAL Workshop, Mumbai. – 2001.

[14] *Dutta S. et al.* Text normalization in code-mixed social media text //Recent Trends in Information Systems (ReTIS), 2015 IEEE 2nd International Conference on. – IEEE, 2015. – С. 378-382.

[15] *Bird S.* NLTK: the natural language toolkit // Proceedings of the COLING/ACL on Interactive presentation sessions. Stroudsburg, USA: Association for Computational Linguistics, 2006. P. 69-72.

[16] *Н. Ю. Шведова (гл. ред.)* Русская грамматика. Т. 1: Фонетика. Фонология. Ударение. Интонация. Словообразование. Морфология.// Наука, 1980. §58-76. Сочетания согласных звуков.

[17] *Валгина Н.С., Розенталь Д.Э., Фомина М.И.* Современный русский язык: Учебник. // Логос, 2002. §70.Произношение сочетаний согласных

[18] *Турдаков Д., Астраханцев Н., Недумов Я., Сысоев А., Андрианов И., Майоров В., Федоренко Д., Коршунов А., Кузнецов С.* Texterra: инфраструктура для анализа текстов //Труды Института системного программирования РАН. – 2014. – Т. 26. – №. 1. – С. 421-437.