УДК: 05.13.06

Р.С. Омельченко

ПРОГРАММА ПРОВЕРКИ ОРФОГРАФИИ (SPELLCHECKER) НА ОСНОВЕ РАСПРЕДЕЛЕННЫХ ПРЕДСТАВЛЕНИЙ

Задача программ проверки орфографии состоит в нахождении и исправлении ошибок в словах. Как правило, программа предлагает пользователю короткий список предполагаемых корректных слов в последовательности от самого вероятного к наименее вероятному. В данной работе исследована возможность применения бинарных распределенных представлений и методов их обработки для представления, поиска и обработки слов с ошибками. Приведены результаты экспериментов на двух наборах слов с типичными орфографическими ошибками, а также проведен сравнительный анализ с другими методами.

Введение

Программа проверки орфографии представляет собой программу, позволяющую находить и исправлять ошибки в словах. Как правило, программа предлагает пользователю короткий список предполагаемых правильных слов в последовательности от самого вероятного к наименее вероятному. Исправление орфографических ошибок в подобных системах может быть организовано одним из таких способов:

– проверка и коррекция ошибок в изолированных словах, при которых каждое слово проверяется отдельно без контекста;

– контекстно-зависимая проверка и коррекция ошибок.

Большинство программ проверки орфографии используют первый вариант, так как второй требует синтаксического анализа и является более сложным и зависящим от конкретного языка. Даже в контекстно-зависимых методах список кандидатов корректных слов получают для изолированных слов, а далее выбор совершается с помощью контекста. В данной работе будет использоваться метод с изолированными словами.

Также использованы распределенные представления, методы их обработки (создания, поиска) для коррекции слов с ошибками. Для оценки и сравнения с другими программами использовались два набора слов с типичными орфографическими ошибками.

Обзор существующих методов

Для сравнения результатов работы нашей программы с другими применялись публично доступные базы, на которых тестировали свои программы проверки орфографии Роджер Миттон [1], а также Себастьян Деоровиц и Марцин Киура [2]. Эти программы не используют информацию о контексте, как и описываемая программа.

Приложение Деоровица и Киуры основывается на правилах, моделирующих типичные орфографические ошибки. Правила применяются к словам с ошибками таким образом, что входная строка, находящаяся где угодно в слове с ошибками, заменяется на выходную строку [2]. Каждой замене на новую строку присвоена стоимость. Если полученное после замены слово корректное, оно добавляется в список кандидатов. Если к слову применяется несколько правил – стоимость суммируется. После этого все кандидаты сортируются в порядке увеличения стоимости. Предполагается, что корректное слово с наименьшей стоимостью является наиболее вероятным кандидатом для исправления слова с ошибкой.

В программе Роджера Миттона [1] использовалась система ключей. Ключи являлись сжатыми версиями слов с ошибкой и слов из словаря, содержащих признаки, наиболее вероятно присутствующие как в слове с ошибкой, так и в оригинале. Ключи строились на основе определенных

правил. Миттоном создан алгоритм, который с помощью этих ключей позволял находить корректные слова. Для сортировки кандидатов использовался метод минимального расстояния коррекции, суть которого состояла в нахождении объема изменений слова с ошибкой для достижения корректного слова из словаря. Для каждой операции коррекции (удаление или вставка буквы, замена одной буквы на другую, перестановка смежных букв) определялась стоимость и вычислялась общая стоимость достижения корректного слова.

Кроме этого данный метод был дополнен методом простого совпадения букв, включающий подсчет несовпадающих букв и дополнительный штраф за несовпадение первой буквы. Чем выше результат, тем хуже совпадение. Также для дополнительной коррекции результата каждого кандидата использовалась частота использования слова (количество появлений в Британской Национальной Коллекции).

Представление слов бинарными распределенными кодвекторами

Бинарные распределенные кодвектора. Основным отличием описываемой программы проверки орфографии от предыдущих программ [1, 2] является представление слов, а также методы проверки. Для представления слов использовались распределенные бинарные разреженные кодвектора.

Распределенное представление информации – форма векторного представления, где каждый объект представлен совокупностью элементов вектора, а отдельный элемент вектора может принадлежать представлениям разных объектов [3]. Такие представления были внедрены в программу проверки орфографии, чтобы использовать их достоинства для проверки корректности слов [3 – 6]. Ведь в самой их природе заложены свойства, позволяющие исправлять ошибки:

– представление релевантных аспектов объектов как элементов многомерного вектора, что обеспечивает непосредственный доступ к ним;

– непосредственное представление сходства – сходные объекты имеют сходные представления. Такое свойство можно использовать для быстрого извлечения похожих представлений из памяти, сравнения с прототипами и автоматического обобщения;

– избыточное представление информации, что приводит к способности работать в условиях шума, сбоев, и неопределенности. Это свойство позволяет производить коррекцию ошибок и восстанавливать информацию по частичному, зашумленному или частично правильному входу;

– надежность и мягкая деградация: информация в представлении все еще доступна, когда части повреждены или отсутствуют. Возможность представления многих объектов на одном множестве элементов;

– возможность вносить малые изменения в содержимое представлений, позволяющая учитывать малые изменения значений и различать незначительные отличия;

– возможность использования методов, развитых в рамках таких направлений, как машинное обучение, линейные модели, распознавание образов и др.;

– нейробиологическая релевантность.

Кроме этого распределенные представления позволяют реализовать алгоритмы проверки слов эффективным образом за счет:

– параллельности алгоритмов обработки, что позволяет эффективную аппаратную реализацию;

– возможности использовать распределенную память, позволяющую упростить процессы хранения и обобщения;

– эффективного использования ресурсов (до 2^N разных состояний для N-мерных бинарных векторных РП).

Представление слов неслучайными кодвекторами. Ввод текста в компьютер сопряжен с наличием возможности совершить ошибку. Задачей программы проверки орфографии является выявление такой ошибки и ее исправление. Выделяют несколько видов ошибок [2]:

- словарная некомпетентность;
- опечатки – ошибки при печатании, такие как вставка лишней буквы, удаление буквы, замена одной буквы на другую, перестановка двух смежных букв;
- неправильное написание – такие ошибки появляются, когда произношение известно, но используются некорректные графемы (буквы представляющие фонемы) для представления фонем из слова (подобные ошибки часто возникают в языках, в которых фонема представлена несколькими графемами).

Как известно, неправильное написание слов более удалено от их оригиналов, чем те ошибки, которые вызваны опечатками [1]. Если опечатки могут быть исправлены уже за счет свойств самих распределенных представлений, то для более сложных ошибок (неправильное написание при известном произношении) представление свойств фонем могло бы улучшить результаты. Однако, учитывая то, что на вход программы проверки орфографии будет подаваться печатный текст, а не звуковая запись, безусловно, нельзя использовать все преимущества представления свойств фонем, так как в некоторых случаях фонема может быть представлена несколькими графемами, а графема в разных словах может означать разные фонемы.

Несмотря на это, учитывая, что чаще всего буква соответствует звуку в устной речи (хотя это и необязательно), мы можем использовать преимущество представления свойств фонем хотя бы частично. Для этого необходимо перенести свойства фонем на буквы, т. е. в большинстве своем при таком представлении, похожие графемы должны отображать похожие фонемы. Такое отображение осуществлено с помощью таблиц соответствия графем фонемам [7].

Учитывая это, в продолжение предыдущих исследований в области представления образов бинарными распределенными кодвекторами, где для представления использовались случайные кодвекторы (для непохожих образов) [8], для данной программы проверки орфографии использовались распределенные представления, формируемые на основе априорных знаний о задаче, а именно используя данные о признаках фонем английского языка. Другими словами, для отображения сходства букв и слов использовано неслучайное представление. Например, графемы «ph» и графема «f» представляют одну и туже фонему /f/ [9]. Соответственно, слова, содержащие эти графемы, должны иметь схожесть кодвекторов.

Преимуществами неслучайного представления является то, что комбинации естественных признаков, повышают их дискриминационные свойства. При признаковом представлении каждый признак обычно характеризует не уникальный объект, а целый ряд объектов (одна и та же частота может быть включена в разные фонемы), а увеличение набора выделяемых признаков позволяет более точно описывать конкретный объект. Кроме этого, другим аргументом в пользу такого подхода является нейрофизиологическая релевантность.

Как известно, связывание контекстно-зависимым прореживанием обеспечивает сохранение информации о группировке компонентов структур [6], а его рекурсивное применение позволяет представлять сложные иерархические структуры. А для направленных отношений (т. е. когда важна последовательность их аргументов) используются схемы роль-заполнитель и предикат-аргументы.

Однако в данной работе для учета последовательности букв, предложен другой метод кодирование слов через кодирования последовательности букв слова.

Безусловно, важная характеристика слова – это последовательность букв в нем, а не только их совокупность. Так, слово «кот» имеет другой смысл, отличный от слова «ток», хотя буквы в них одинаковые и звучание схожее. Соответственно, им должны отвечать различные кодвектора, учитывающие различные последовательности букв в слове. Такие отношения вида R(A,B,...) (где R – идентификатор отношения, A,B... – аргументы), у которых важна последовательность их аргументов, являются направленными.

Отличие слогов «от» от «то» с точки зрения нейрофизиологии в том, что в

головном мозге возбуждаются разные ансамбли нейронов. Учитывая это необходимо формировать различные кодвекторы для разных последовательностей одинакового набора букв.

Учитывая этот факт, а также по аналогии с наличием «обратных связей» в нервной системе человека, в работе был предложен новый метод формирования кодвекторов таких отношений.

Для этого использовалось кодирование биграмм. Для каждой графемы выделено поле вектора. Это поле делилось на поля связей с другими буквами. Количество таких полей равнялось количеству букв в алфавите. Таким образом, каждой биграмме присваивался уникальный кодвектор. Формирование кодвекторов происходило таким образом, что единицы в векторе располагались в виде концентрированных групп – ансамблей (табл. 1 и 2).

Таблица 1. Биграмма в слове

Биграмма	Слово, содержащее биграмму
Bi	Bit

Таблица 2. Представление биграмм

Позиция бита	1	…	1822	1823	…
Кодвектор	0	…	0	1	…

После того, как каждой биграмме был присвоен кодвектор, следующей задачей являлось формирование словаря, а именно бинарных кодвекторов, которые соответствовали бы словам. Для кодирования слов вектора биграмм соединялись по дизъюнкции. При этом для построения слова использовались биграммы соседних букв в прямом направлении, а также все возможные варианты биграмм в обратном направлении, замыкая связи между буквами в круг.

Например, в слове «came» по дизъюнкции объединялись вектора 9 биграмм (рис. 1): 3 прямых: «ca», «am», «me» (черные стрелки) и 6 обратных (синие стрелки): «ac», «ma», «mc», «ec», «ea», «em».

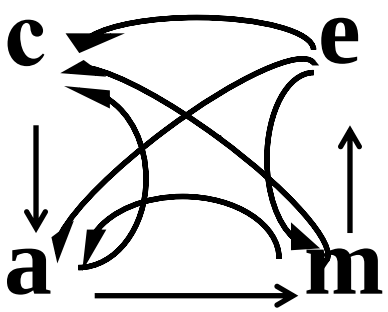

Рис. 1. Кодирование последовательности букв

Кроме этого, чем дальше расположена в слове буква относительно второй буквы биграммы, тем меньше количество единиц в векторе этой биграммы. Таким образом, кодировалось относительное положение букв в слове. Учитывая эксперименты, проведенные ранее [10], размер кодвектора принят n = 18 000 (табл. 3).

Таблица 3. Построение вектора слова с помощью дизъюнкции

Буква	Кодвектора биграмм / слова							
BI	0	…	0	0	0	…	0	0
IT	0	…	1	1	1	…	0	0
IB	0	…	0	0	0	…	1	0
TI	0	…	0	0	0	…	0	0
TB	0	…	0	0	0	…	0	1
BIT	0	…	1	1	1	…	1	1

Сравнение слов и поиск корректного слова. После того, как формируется кодвектор слова, он записывается в файл, так называемый словарь. Для поиска кодвекторов слов использовался метод поиска ближайших аналогов.

Поиск наиболее близкого аналога в памяти сводится к нахождению (в памяти, где хранятся кодвекторы слов) кодвектора, наиболее похожего на входной.

Поиск ближайших аналогов осуществляется по величине разницы перекрытия единиц и разных битов их кодвекторов из базы $\mathbf{X}_l$ с кодвектором входного аналога $\mathbf{X}_{вх}$:

$$l^*(x_{вх}) = \arg\max_{l=1,L}\left(V(X_{вх}, X_l) - Z(X_{вх}, X_l)\right),$$

где l = 1, L – индекс кодвектора в базе; L – число аналогов (эпизодов, слов и т. п.) в БЗ; V(.,.) – величина перекрытия кодвекторов; Z(.,.) – количество отличающихся битов кодвекторов.

В задаче поиска ближайших аналогов для нахождения величины сходства кодвекторов применен метод обратного индексирования [11].

Таким образом, для каждого кандидата (вектора из словаря) вычисляется величины сходства, на основе которых вектора-кандидаты можно упорядочить. Определение величин сходства входного кодвектора с кодвекторами *L* эпизодов базы (слов) осуществляется путем формирования выходного *L*-мерного вектора сходств s, элементы которого – величины сходства. Элементы вектора s формируются следующим путем:

а) в его элементы прибавляются единицы для номеров кодвекторов базы, содержащихся в строках, соответствующих ненулевым элементам входного кодвектора – $X_{вх}$;

б) итоговое значение каждого элемента рассчитывается по формуле:

$$\theta_l = 3^* V_l - X_l - X_l \ ,$$

где $|\mathbf{X}_{вх}|$ – количество единиц во входом векторе; $|\mathbf{X}_l|$ – количество единиц в кодвекторе эпизода базы; $\mathbf{V}_l$ – величина перекрытия кодвекторов. Вектор с наибольшей величиной сходства должен быть наиболее подходящим кандидатом и соответственно, вектором слова, которое требовалось найти. Для дальнейшего сравнения с другими системами формировался список из 10-и наиболее похожих векторов-кандидатов на входной.

Результаты экспериментов

Два набора слов, содержащих реальные орфографические ошибки, которые использовались для экспериментов: aspell и wikipedia. Тестовые данные – это простой список слов с ошибками и соответствующих им корректных слов. Первая база – aspell – это коллекция ошибок, которые сложно корректируются, которая использовалась для тестирования GNU Aspell программы проверки орфографии [12]. Она содержит 525 слов с ошибками – как опечатки (olf для old), так и неправильное написание (funetic для phonetic).

Другая база – Wikipedia – это набор типичных орфографических ошибок, которые сделали пользователи Википедии. Эта база больше, она содержит в основном опечатки или незначительные ошибки [13].

Для сравнения мы использовали формат базы, который переписан Миттоном (табл. 4) [1].

Таблица 4. Выдержка из базы ошибок «Wikipedia»

$Apennines
Apenines
Appenines
$Athenian
Athenean
$Athenians
Atheneans
$Bernoulli
Bernouilli
$Blitzkrieg
Blitzkreig
$Brazilian
Brasillian

Данная база содержит 2455 ошибок. Корректные формы слов переписаны Миттоном согласно Британского английского языка. Как и у Деоровица с Киурой и у Миттона, ошибки, корректные формы которых не содержал словарь, исключались из анализа.

Для вычисления значений лучших кандидатов, использовалась следующая формула, определяющая точность определения корректных слов-кандидатов:

$$Top - n = \frac{t_n}{t_t} * 100\%,$$

где t_n – найденные корректные слова, которые содержит список из n слов-кандидатов, t_t – количество слов с ошибками в базе, для $1 \le n \le 10$.

Пример найденного списка наиболее подходящих кандидатов приведен в табл. 5:

Таблица 5. Слова – кандидаты

Входное слово с ошибкой	Слова – кандидаты в порядке убывания величины сходства	Корректное слово из базы «Aspell»
Abilitey	Ability, liability, viability, affability, stability, inability, abilities, tenability, amiability, habitability.	Ability
Beging	Begging, begin, beginning, begins, egging, beginner, beggings, pegging, besieging, began.	Beginning

Сравнения результатов нескольких программ обработки орфографии приведены в табл. 6 и 7.

Таблица 6. Сравнение результатов программ проверки орфографии (база aspell)

	aspell Деоровици Киура	aspell Миттон	aspell (эта программа)
First	66,3%	71,1%	58,6%
Top two	75,5%	83,2%	69,7%
Top three	79,6%	88,6%	77,7%
Top five	83,6%	91,4%	82,4%
Top ten	85,5%	94,4%	88,9%
Total = 100%	511	499	488

Таблица 7. Сравнение результатов программ проверки орфографии (база wikipedia)

	wikipedia Деоровици Киура	wikipedia Миттон	wikipedia (эта программа)
First	94,1%	92,9%	80,0%
Top two	97,4%	97,2%	89,4%
Top three	98,3%	97,9%	92,8%
Top five	98,9%	98,6%	95,7%
Top ten	99,0%	99,0%	97,5%
Total = 100%	2196	2154	2284

Из табл. 6 видно, что для набора слов aspell частота попадания правильного слова в десятку первых кандидатов у предложенного метода лучше, чем у Деоровица и несколько хуже, чем у Миттона. Для словаря Wikipedia этот показатель также довольно близок показателям сравниваемых программ. Несколько хуже в сравнении результаты работы рассматриваемой программы для правильного слова – первого кандидата, однако необходимо учесть, что в данном подходе не разрабатывались специальные правила для конкретных языков, а использовались существующие методы на основе бинарных распределенных представлений. Дальнейшего улучшения результатов можно ожидать при внедрении в анализ синтаксиса предложения и семантики. Именно в этом направлении будет развиваться дальнейшая работа.

Заключение

В данной работе предложен подход к проверке орфографии на основе распределенных представлений. Для этого создан словарь разреженных бинарных распределенных векторов для каждого слова английского языка.

Как проверка корректности слова, так и его коррекция в случае, если в нем есть ошибки, зависят от размера словаря. Объем словаря, который использовался в данной работе – 58112 слов. Для коррекции слова необходимо вычислить количество общих единиц вектора входящего слова с каждым вектором словаря с помощью метода обратного индексирования. Некоторые программы предполагают, что первая буква слова с ошибкой корректна, что, как правило, так и есть [14], при этом они ограничивают поиск словами, которые начинаются с той же буквы, что и слово с ошибкой. Однако в данной работе поиск происходит по всем словам словаря.

Программа проверки орфографии на основе распределенных представлений может использоваться для любых языков. При чем, если у других программ проверки орфографии необходимо собрать набор корректных слов и создавать новый

набор правил замен [2] или отдельно адаптировать под каждый язык [1], то в данной программе достаточно присвоить вектора буквам алфавита (только новым графемам) и в автоматическом режиме изучить (присвоить бинарные вектора словам словаря) лексикон. Другими словами фактически отсутствует необходимость или необходима минимальная адаптация к другим языкам.

Еще одним преимуществом по сравнению с другими программами проверки орфографии является то, что у них правила или методы подгоняются под конкретные множества ошибок, т. е. существует тестовое множество, на котором эти методы тестируются. В нашей программе тестовое множество ошибок не используется, а коррекция происходит на основе представления содержимого словаря и методов обработки бинарных распределенных векторов.

В отличие от других программ проверки орфографии пользователи данной программы могут расширить лексикон или вовсе использовать свой собственный. Например, с лексиконом ADFA, который использует Деоровиц [2], это невозможно.

Как известно, неправильное написание слов орфографически более удалено от их оригиналов [1], чем те ошибки, которые вызваны опечатками. В таких случаях часто правильные фонемы заменяются на близкие по звучанию. Вышеописанное представление слов позволяет учесть в какой-то мере схожесть фонем, а значит, имеет больше шансов исправить более сложные ошибки, которые возникают вследствие того, что кто-либо не уверен в написании или произношении слова.

Кроме того, в распределенном представлении слов, а также благодаря методам связывания букв существует аналогия с некоторыми правилами, которые специально разрабатывались в других программах распознавания орфографии. Например, алгоритм Миттона [1] предполагает, что для ранжирования кандидатов подсчитываются количество несовпадающих букв и пар букв. В данной программе также вектора слов включают вектора биграмм, а сумма сравнение векторов учитывает как совпадающие единицы векторов, так и те, которые не совпадают. Соответственно, чем больше совпадающих букв и чем меньше несовпадающих букв, тем более похоже слово.

Результаты, описанные в данной статье, показывают возможность применить распределенные представления для исправления орфографических ошибок в словах, а также при этом получить качественные результаты. Однако, недостаток такой системы – это обработка изолированных слов. Тем не менее, в дальнейшем возможности таких программ можно расширить, внедряя в такие вектора слов синтаксические и семантические признаки. Причем в данном случае такие признаки могут помочь в улучшении как выбора круга наиболее вероятных кандидатов, так и в их более корректном ранжировании.

1. *Roger Mitton.* Ordering the suggestions of a spellchecker without using context // Natural Language Engineering, 2009. – 15 (2). – P. 173–192.
2. *Sebastian Deorowicz, Marcin G. Ciura.* Correcting Spelling Errors by Modeling their Causes // International Journal of Applied Mathematics and Computer Science, 2005. – 12(2). – P. 275–285.
3. *Thorpe S.* Localized Versus Distributed Representations // Arbib M. The Handbook of Brain Theory and Neural Networks. – Cambridge, MA: MIT Press, 2003. – P. 643–646.
4. *Browne A., Sun R.* Connectionist Inference Models // Neural Networks. – 2001. – Vol. 14, N 10. – P. 1331–1355.
5. *Eliasmith C., Thagard P.* Integrating Structure and Meaning: A Distributed Model of Analogical Mapping // Cognitive Science. – 2001. – Vol. 25, N 2. – P. 245–286.
6. *Rachkovskij D.A.* Representation and Processing of Structures with Binary Sparse Distributed Codes // IEEE Transactions on Knowledge and Data Engineering. – 2001. – Vol. 13, N 2. – P. 261–276.
7. *Letters* and Sounds: Principles and Practice of High Quality Phonics Notes of Guidance for Practitioners and Teachers Andrew Adonis Rt Hon Beverly Hughes 11 May 2010. Available at http://www.education.gov.uk/

8. *Rachkovskij D.A., Kussul E.M.* Binding and Normalization of Binary Sparse Distributed Representations by Context-Dependent Thinning // Neural Computation. – 2001. – Vol. 13, N 2. – P. 411–452.
9. *Nima Mesgarani, Stephen David, Shihab Shamma.* «Representation of phonemes in primary auditory cortex: how the brain analyzes speech», Electrical and Computer Engineering Department University of Maryland, College Park, MD 20742, ICASSP, 2007.
10. *Rachkovskij D.A., Slipchenko S.V.* Similarity-Based Retrieval with Structure-Sensitive Sparse Binary Distributed Representations // Computational Intelligence. –2012. – Vol. 28, N 1. – P. 106–129.
11. *Слипченко С.В., Рачковский Д.А., Мисуно И.С.* Декодирование разреженных бинарных распределенных кодов скалярных и векторных величин // Компьютерная математика. – 2005. – № 3. – С. 108–120.
12. *Atkinson K.:* Spell Checker Test Kernel Results – 2011. – Available at http://aspell.net/test/cur/
13. *Wikipedia* Corpora of misspellings. – Available at http://www.dcs.bbk.ac.uk/~roger/wikipedia.dat.
14. *Yannakoudakis E.J. and Fawthrop D.* The rules of spelling errors // Information Processing and Management. – 1983. – N 19(2). – P. 87–99.

Получено 18.02.2013

Об авторе:

Омельченко Руслан Сергеевич,
аспирант.

Место работы автора:

МННЦІТіС,
(067) 378 4552,
E-mail:
ruslan.omelchenko.irtcits@gmail.com.