Обзор методов предиктивного ввода текста

Филатов С.Ю., МГТУ им. Н.Э. Баумана
serj.phil@outlook.com

Аннотация

Данная работа посвящена проблеме автодополнения текста на русском языке. Проводится обзор существующих решений, выявляются их преимущества, недостатки и применимость к поставленной задаче. Предлагается улучшение алгоритма на основе N-грамм, повышающее качество его работы для русского языка.

1 Актуальность проблемы

Автодополнение и предиктивный ввод текста предназначены для увеличения скорости и облегчения набора текста. Необходимость в наличии и развитии данных средств резко выросла с появлением мобильных телефонов и устройств с сенсорным экраном, на которых является затруднительным быстрый и безошибочный набор текста.

Существующие подходы к решению проблемы автодополнения текста основаны на использовании словарей, содержащих слова и вероятность их встречи в тексте. Данный подход показывает свою эффективность при использовании с аналитическими языками, характеризуемыми малым числом словоформ. Однако при использовании данного метода автодополнения текста с синтетическими флективными языками (например, русским), имеющими большое число словоформ, качество предсказания падает, поскольку метод не учитывает морфологические параметры вводимых слов. Частично данная проблема может быть решена путём использования биграмм, но значительный рост размера словаря мешает использованию биграмм для повышения качества предсказания.

В данной работе будет проведён анализ возможных путей решения упомянутых проблем с учётом фактора производительности и будет предложен подход, который позволит повысить качество автодополнения текста на флективном русском языке.

2 Обзор существующих методов автодополнения текста

Задача автодополнения вводимого текста состоит в предсказании вводимого пользователем слова по его началу.

Назовём классом согласования словоформы совокупность её граммем, определяющих либо определяемые значениями граммем словоформ, синтаксически связанных с данной. Рассмотрим возможные классы согласования на примере наречий и прилагательных.

Наречия, вступая в подчинительную связь с другими частями речи, не изменяют свою форму и, следовательно, относятся к одному классу согласования.

Прилагательные, вступая в подчинительную связь, согласуются с главным словом в трёх грамматических категориях: роде, числе и падеже. Таким образом, число классов согласования прилагательных определяется числом различных наборов значений категорий рода, числа и падежа.

Для остальных частей речи число классов согласования определяется по аналогии.

2.1 Стохастические методы

Для решения задачи автодополнения обычно используются стохастические методы.

Для предсказания окончания одного слова используется методы, основанные на использовании марковских моделей [Daniel, James, 2000].

Для предсказания последовательностей слов используется модели, основанные на N-граммах.

Для определения наиболее вероятного продолжения последовательности w_n слов $w_1 \dots w_{n-1}$ используются вероятности встречи последовательности $w_1 \dots w_n$:

$$P(w_n|w_1 \dots w_{n-1}) = \frac{P(w_1 \dots w_n)}{\prod_{i=1}^{n-1} P(w_1|w_1 \dots w_{i-1})}.$$

К сожалению, собрать статистику взаимного употребления длинных последовательностей слов довольно

непросто. Это потребует наличия большого корпуса исходных текстов и значительного объёма памяти для хранения собранной статистики. Поэтому используется упрощённая модель, в которой считается, что вероятность того, что слово w продолжит последовательность $w_1 \ldots w_n$ зависит только от N предыдущих слов в последовательности.

$$P(w_n|w_1 \ldots w_{n-1}) \approx P(w_n|w_{n-N} \ldots w_{n-1}).$$

Из-за большого объёма необходимой для хранения N-грамм памяти на практике используются смешанные модели, использующие униграммы для всех слов языка и биграммы для пар наиболее часто встречаемых слов.

Для языков флективного строя использование такой модели может привести к резкому снижению качества предсказания за счёт потери согласованности по морфологическим параметрам пар слов, не входящих в группу наиболее распространённых.

Помимо этого, следует обратить внимание на особенность работы механизмов предиктивного ввода, основанных на использовании N-грамм. Поскольку метод учитывает частоты взаимных употреблений лексем, в результате работы механизма будут предлагаться варианты завершения слов, соответствующие известным механизму шаблонным фразам.

2.2 Метод commonsense

Метод commonsense [Stocky, Faaborg, Lieberman, 2004] похож на метод биграмм, однако имеет существенные отличия в механизме получения возможных продолжений вводимой фразы и метода оценки вероятности встречи предлагаемых вариантов.

Вместо использования словаря метод commonsense предполагает использование базы знаний (семантической сети, содержащей знания в виде утверждений «теннис — это вид спорта» и «для игры в теннис требуется ракетка»).

После ввода слова w получает из базы знаний семантический контекст слова w и присваивает каждому слову в утверждении (где n – порядковый номер утверждения) вес:

$$\text{score} = \frac{1}{\log_5(1+n)}.$$

Основание логарифма выбрано экспериментальным путём [Stocky, Faaborg, Lieberman, 2004].

Из полученных пар «словоформа — вес» строится словарь, используемый для предложения вариантов следующего слова во вводимой пользователем фразе.

Результаты исследований [Stocky, Faaborg, Lieberman, 2004] показывают небольшое преимущество данного метода над статистическими (рис. 1), однако метод не является адаптированным для использования с языками флективного строя.

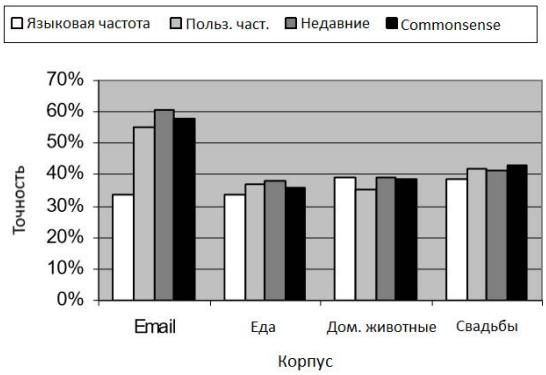

Рис. 1. Сравнение точности предсказания методом commonsense и различными статистическими методами

2.3 Метод reflective text

Метода reflective text [Sandnes, 2015] реализует предсказание окончания вводимого слова путём предсказания раскрытия введённого пользователям сокращения. Для работы метода используется словарь.

Для начала работы метода требуется ввод трёх и более символов. Введённые символы сравниваются со словами словаря с использованием алгоритма наибольшей общей подпоследовательности (LCS) [Hunt, Szymanski, 1977]. В качестве возможных завершений предлагаются слова словаря, содержащие запрошенную последовательность. Если пользователь явно не выбрал правильное завершение вводимой им последовательности символов, система использует кратчайший из вариантов. При наличии нескольких вариантов предпочтение отдаётся тем, что начинаются с введённой пользователем последовательности.

Рассмотрим работу алгоритма для норвежского языка (рис. 2).

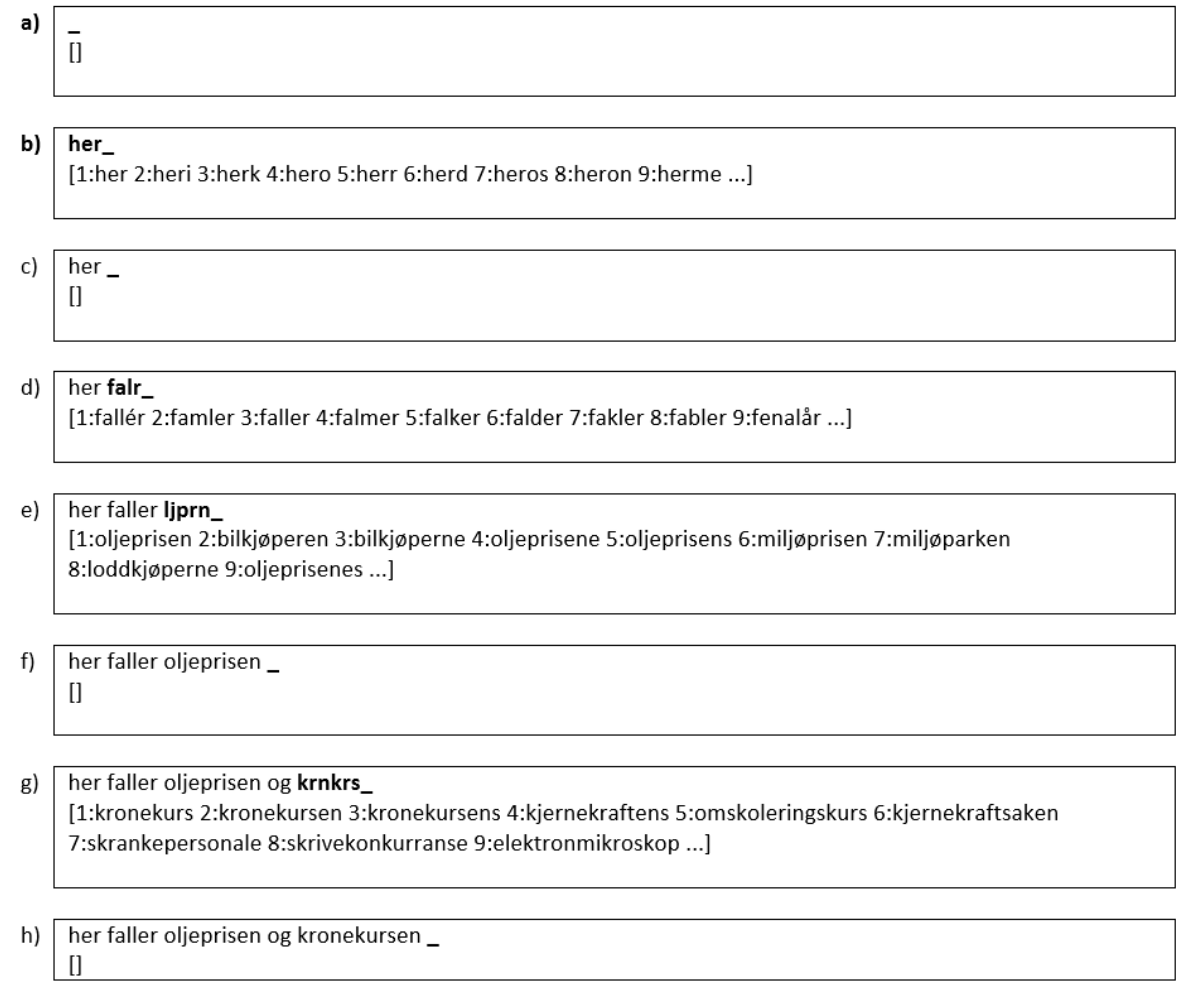

Рис. 2. Пример **работы** метода reflective text

Пользователь собирается ввести фразу «her faller oljeprisen og kronekursen» (падают цены на нефть и курс кроны).

До начала ввода под курсором отображаются пустые квадратные скобки, обозначающие отсутствие предлагаемых вариантов.

Пользователь вводит слово «her» и видит список предлагаемых вариантов.

Слово «her» является первой альтернативой и будет выбрано при нажатии клавиши «пробел».

Пользователь вводит последовательность «falr».

При нажатии клавиши «3» последовательность «falr» заменяется на слово «faller» и вводится пробел.

Пользователь вводит последовательность «ljprn».

Слово «oljeprisen» является первой альтернативой и будет подставлено при нажатии на клавишу «пробел».

Пользователь вводит слово «og» и последовательность «krnkrs».

При нажатии на клавишу «2» последовательность «krnkrs» заменяется на слово «kronekuersen» и вводится пробел.

Исследования производительности [Sandnes, 2015] были проведены для словаря английского языка, содержащего 45374 словоформы, и словаря норвежского языка, содержащего 181731 словоформу. Для обработки одного нажатия на клавишу и подготовки списка возможных вариантов раскрытия сокращения в среднем потребовалось 109 мс (среднеквадратичное отклонение 67.4) для словаря английского языка и 395 мс (среднеквадратичное отклонение 379.5) для норвежского языка.

3 Выбор наиболее подходящего алгоритма для русского языка

Рассмотренные выше алгоритмы обладают определёнными преимуществами и недостатками.

К достоинствам алгоритма, основанного на использовании N-грамм, стоит отнести

простоту его реализации, приспосабливаемость к любым языкам, выдачу семантически подходящих вариантов, возможность обучения метода на любом корпусе. К недостаткам данного метода относится значительный объём памяти, необходимой для хранения словарей.

Метод commonsense представляет интерес, однако используемая в нём база знаний обладает ограниченным размером и не может быть пополнена пользователем.

Практическая применимость метода reflective text вызывает существенные сомнения, поскольку необходимость постоянного выполнения поиска возможных вариантов методом LCS вызывает существенные проблемы с производительностью.

На основании вышесказанного считается практичным производить разработку метода автодополнения текста на русском языке на основе метода N-грамм.

4 Структуры данных, используемые в методах автодополнения

Для представления словаря в памяти возможно использование одной из следующих структур данных:

- таблиц;
- конечных автоматов;
- префиксных деревьев.

Наиболее простым является представление словаря в виде пар (слово, вероятность встречи). К недостаткам данного метода относится большое время поиска наиболее вероятного продолжения для заданной строки и большой занимаемый объём памяти.

Представление словаря в виде конечного автомата позволяет значительно уменьшить потребление оперативной памяти и позволяет быстро проверять наличие слова в словаре, но в связи с особенностями реализации словаря невозможно хранение различной информацией, связанной с отдельными словами в словаре. Данный недостаток связан с тем, что при использовании конечного автомата допускается наличие общего суффикса у элементов словаря, имеющих различный префикс. Эта особенность данной структуры данных позволяет значительно сократить требуемый объём памяти, но приводит к потере информации о пути, по которому была достигнута конечная вершина автомата, что делает невозможным размещение в ней информации, уникальной для каждой словоформы.

Наиболее оптимальными структурами для представления словаря при решении задачи автодополнения текста являются префиксные деревья. Префиксные деревья требуют меньшего объёма памяти, чем таблицы, позволяют осуществлять поиск наиболее вероятного окончания слова за минимальное время и дают возможность хранить различную информацию о слове. Для увеличения эффективности хранения данных можно использовать различные виды префиксных деревьев [Hsu, Ottaviano, 2013].

5 Существующие механизмы автодополнения

Рассмотрим существующие реализации методов автодополнения текста на примере latin input method в операционной системе Android.

Latin input method представляет собой виртуальную клавиатуру с функцией предиктивного ввода, используемую для языков с консонантными и консонантно-вокалическими алфавитами.

Для знакомства с реализацией механизма автодополнения проведём анализ исходного кода пакета LatinIME [Исходный код ОС Android].

Для представления словаря в оперативной памяти используется префиксное дерево. Latin input method позволяет использовать для предсказания пользовательского ввода модель, использующую униграммы и биграммы. Однако для уменьшения объёма требуемой памяти биграммы используются только для наиболее распространённых пар словоформ. Метка bigram элемента словаря используется для изменения веса слова, имеющего эту метку, при выборе наиболее вероятных слов, следующих за данным. В качестве единиц словаря используются словоформы. Исправление наиболее распространённых орфографических ошибок реализовано путём подмены слова с ошибкой на слово без ошибки при предсказании ввода.

Возможности автодополнения вводимого текста имеются и в других операционных системах и коммерческих продуктах, однако исследование алгоритмов их работы невозможно из-за лицензионных ограничений.

6 Описание предлагаемого метода автодополнения

Метод автодополнения вводимого текста, рассматриваемый в данной работе, использует данные о взаимном употреблении классов согласования для повышения качества работы. Рассмотрим стандартные алгоритмы автодополнения с использованием униграмм и биграмм (рис. 3).

Из схемы алгоритма очевидно, что стандартный алгоритм автодополнения, основанный на использовании униграмм, не может обеспечить согласование предлагаемых вариантов завершения слова с ранее введённым текстом, а алгоритм, основанный на биграммах, обеспечивает согласование за счёт запоминания механизмом автодополнения шаблонных пар слов.

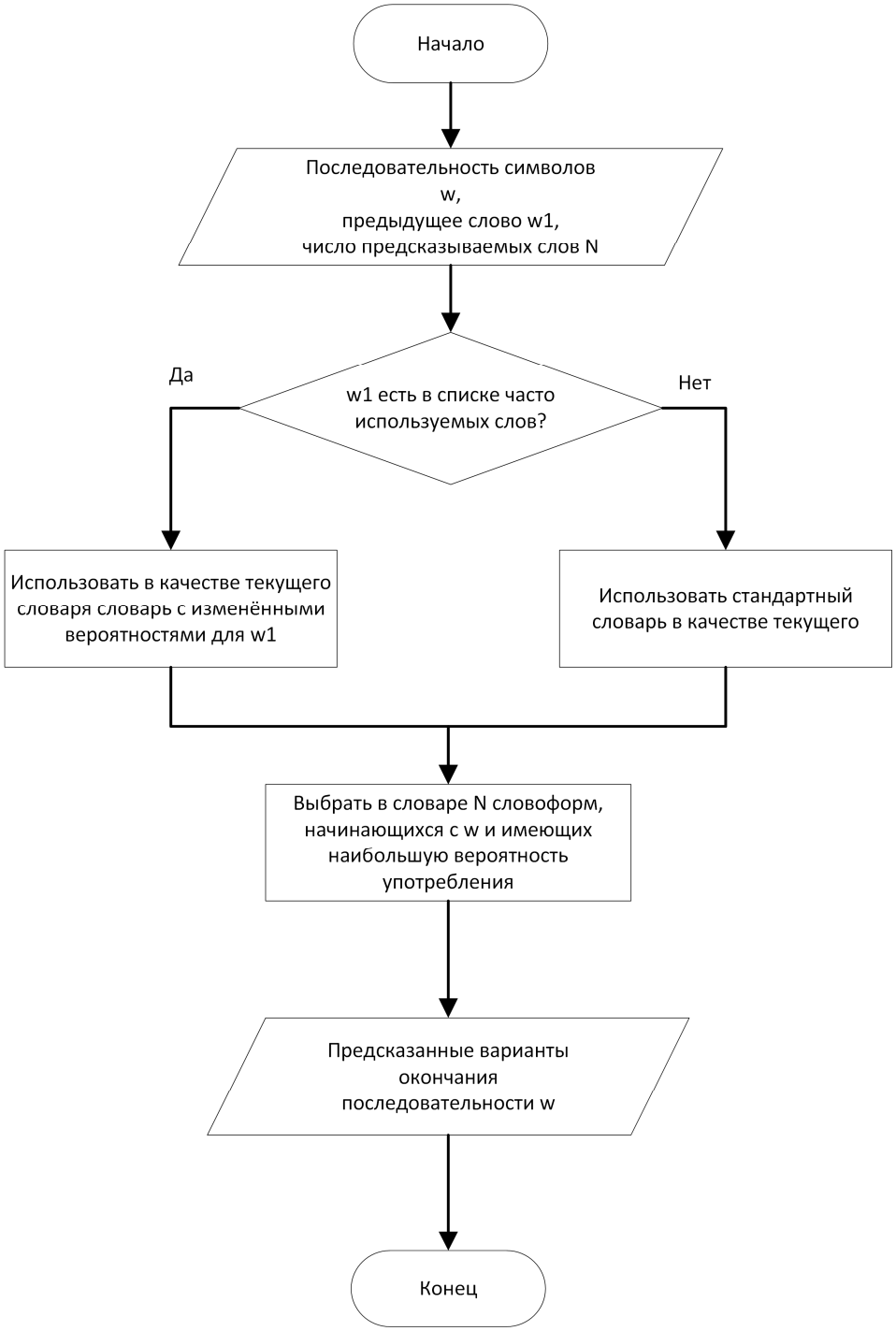

Рис. 3. Схема алгоритма автодополнения с использованием униграмм и биграмм

Для решения данной проблемы предлагается внести в алгоритм следующие изменения.

Всем словоформам, хранящимся в словаре, добавляется свойство, указывающее класс согласования, к которому относится данная словоформа.

В механизм автодополнения вводится таблица частот взаимного употребления классов согласования словоформ.

В процессе обучения и формирования словаря словоформа относится к одному классу согласования (нескольким в случае омонимии) и учитывается в таблице взаимного употребления классов согласования.

Сам алгоритм выбора возможных окончаний вводимой последовательности символов изменится и примет вид, представленный на рис. 4.

Внесённое изменение исключает из рассмотрения алгоритмом автодополнения слова, относящиеся к классам согласования, появление которых в связке с классом

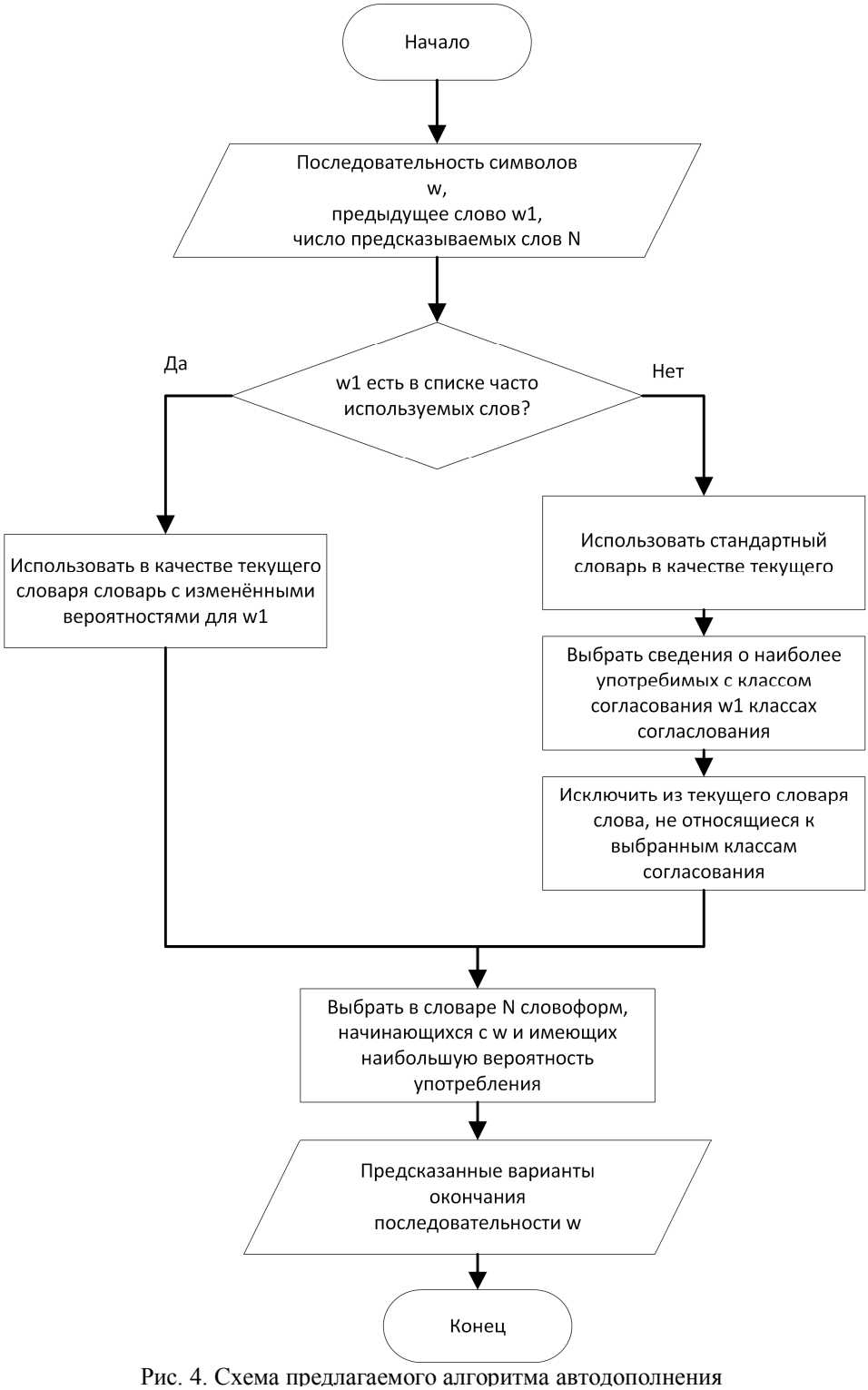

Рис. 4. Схема предлагаемого алгоритма автодополнения

согласования последнего введённого слово маловероятно. Это приводит к грамматическому согласованию предлагаемых вариантов окончания последовательности символов с уже введёнными словами.

Заключение

В данной работе были рассмотрены существующие методы автодополнения вводимого текста. Выявлены достоинства и недостатки рассмотренных методов применительно к задаче для флективного языка. Изучены существующие программные средства автодополнения вводимого текста. Предложены способы улучшения качества работы метода, основанного на использовании N-грамм, для русского языка.

Список литературы

Martin J. H., Jurafsky D. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.* – Prentice Hall, 2000.

Stocky T., Faaborg A., Lieberman H. *A commonsense approach to predictive text entry* //CHI'04 Extended Abstracts on Human Factors in Computing Systems. – ACM, 2004. – Pp. 1163-1166.

Sandnes F. E. *Reflective text entry: a simple low effort predictive input method based on flexible abbreviations* //Procedia Computer Science. – 2015. – T. 67. – Pp. 105-112.

Hunt J. W., Szymanski T. G. *A fast algorithm for computing longest common subsequences* //Communications of the ACM. – 1977. – V. 20. – №. 5. – Pp. 350-353.

Hsu B. J. P., Ottaviano G. *Space-efficient data structures for top-k completion* //Proceedings of the 22nd international conference on World Wide Web. – ACM, 2013. – Pp. 583-594.

Исходный код ОС Android. URL: https://github.com/CyanogenMod/android_packages_inputmethods_LatinIME (дата обращения 20.02.2017).