А.В.Сиренко

ФОРМАЛЬНО-ЛОГИЧЕСКАЯ МОДЕЛЬ ФУНКЦИОНИРОВАНИЯ КОГНАЙЗЕРА[1]

Формальной основой вычислительной модели когнайзера является взвешенная контекстно-зависимая грамматика, интегрирующая различные лексикографические объекты. Традиционным механизмом срабатывания правила грамматики является совпадение левой части правила с последовательностью предложения в грамматике. Учитывая размер алфавита грамматики, сложно составить множество правил, которые будут значимы при обработке запроса. Необходимо выработать признаки применения правила, допускающие неполное совпадение левой части правила и подстроки. За основу таковых предлагается принять оценку редакционного расстояния между некоторой подстрокой пропозиции и левой части правила. Подстрока берется как целостный участок, разрывы не допускаются.

Требуется: для строки-источника определить все вхождения строки-шаблона с допуском несовпадающих символов и количественной оценкой совпадения.

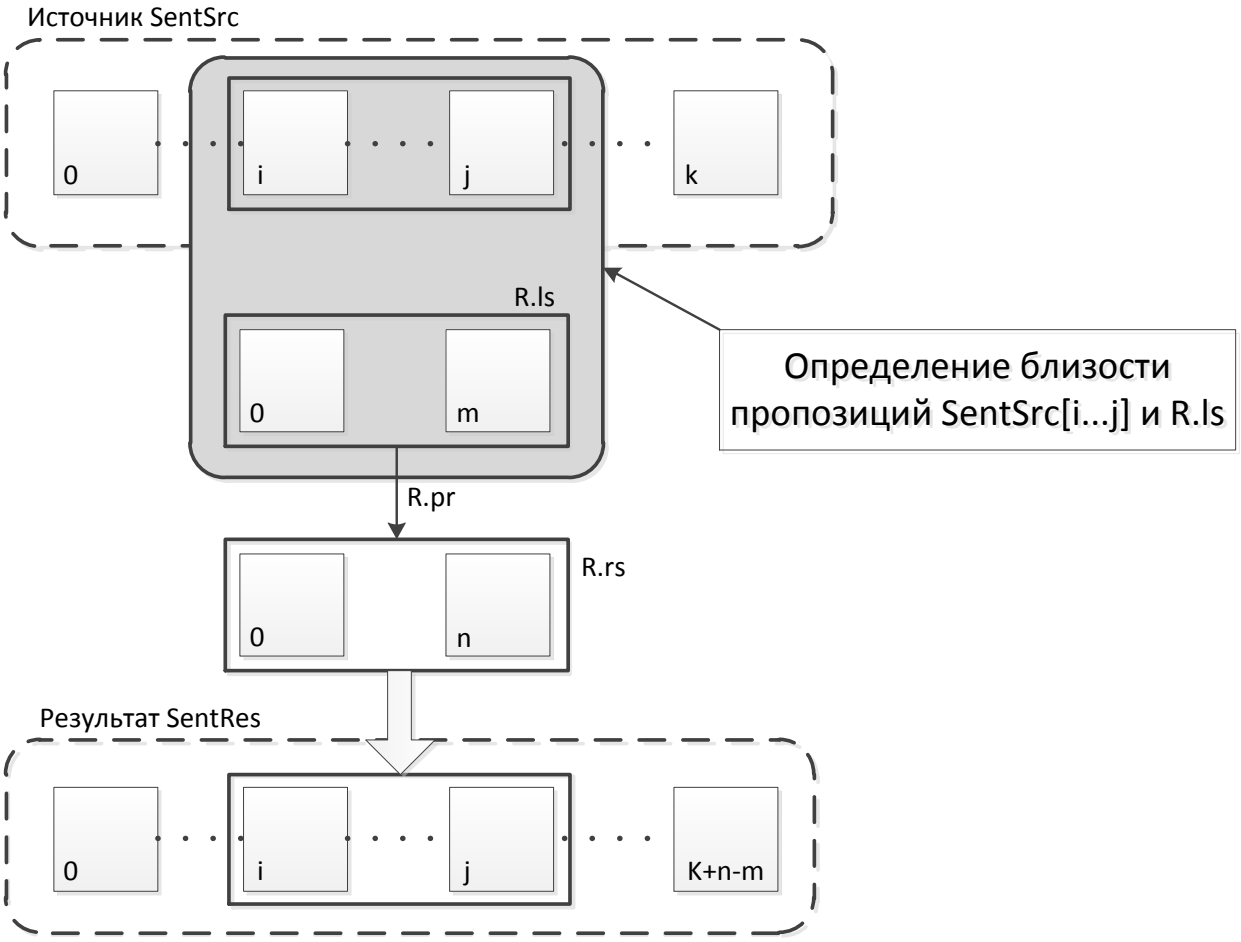

Рисунок 1. Применение правила общего вида.

[1] В данной статье приводятся результаты исследований, выполненных при поддержке гранта РГНФ №12-04-12039в

Определение близости подстроки исходной пропозиции $SentSrc_{i..j}$ и строки R.ls является задачей нечеткого поиска в тексте шаблона. Существует большое число алгоритмов, предназначенных для этой цели, однако они по-разному представляют отличия шаблона и источника. Приведем список параметров с соответствующими нашей задаче значениями (Таблица 1).

№	Условие применения алгоритма	Актуальное значение
1.	Соотношение длин источника (SourceSize) и шаблона (PatternSize): поиск подстроки, либо анализ соответствия строк.	SourceSize произвольно соотносится с PatternSize
2.	Точное совпадение, либо допускаются отличия	Допускаются отличия
3.	Длина текста для поиска и длина шаблона	SourceSize = 1..30, PatternSize = 1..30
4.	Вычислительные ресурсы на выполнение поиска и предобработку данных: память и процессорное время.	Количество шаблонов для каждой строки-источника имеет порядок 10^4 ($PatternSize > 1$).
5.	Качественное или количественное определение вхождение шаблона в текст.	Требуется количественное значение, пропорциональное числу общих компонент и порядку слов шаблона и источника.
6.	Расширение шаблона с помощью групповых символов (wildcard, символ-джокер), вплоть до регулярных выражений.	Требуется для допуска разрывов в шаблоне или поиска по части речи.
7.	Поведение алгоритма при изменении допустимого числа отличий (при поиске неполных совпадений).	Число допустимых отличий зависит от их характера, до половины SourceSize.
8.	Сложность реализации, наличие готовых компонент.	Использование готовых компонент предпочтительно, при способности их адаптации к задаче.

Таблица 1. Условия выбора алгоритма сравнения строк.

Проанализируем требования к алгоритму.

Пункты 1,2. Соотношение длин источника и шаблона требует от алгоритма: анализ сходства строк при совпадении длин, либо длине шаблона более длины строки; поиск подстроки при длине источника большей, чем длина шаблона. Данному требованию удовлетворяют алгоритмы поиска шаблона в тексте с допустимым неполным совпадением.

Пункты 3,4. Из размеров источника и шаблона, а также требования работы с множеством шаблонов для каждого источника следует, что возможно использовать сложную предобработку источника с построением дополнительных структур. Предобработка шаблона нежелательна.

Пункты 5,6. Из необходимости определения количественной характеристики следует недостаточность традиционных алгоритмов поиска совпадений с ошибками, допускающими ограничение по числу несовпадений. Использование групповых символов требует простоты алгоритма к расширению функциональных возможностей.

В результате анализа ряда источников, в числе которых [Navarro, 2001; Гасфилд, 2003, гл.5,6,11; Кормен и др., 2005, гл.32], был спроектирован недетерминированный автомат нечеткого поиска подстрок. Автомат формируется на основе суффиксного дерева предложения (алгоритм Укконена) и редакционных расстояний Дамерау-Левенштейна. В

качестве параметров оптимизации выступают координаты подстроки источника и редакционное предписание. Целевой функцией является редакционное расстояние.

1. Свойства редакционных операций.

Строка-источник при построении суффиксного дерева известна полностью, поэтому отвергаем алгоритмы построения суффиксных деревьев on-line (по мере поступления символов строки) как более ресурсоемкие и воспользуемся традиционным порядком построения несжатого суффиксного дерева. Суффиксное дерево должно быть несжатым, что будет востребовано при вычислении редакционных расстояний.
Количественная оценка близости строк должна проводиться с учетом:

a) пропорционально числу совпадающих элементов и в соответствии с их типом;

b) соответствия порядка строк – несовпадение порядка снижает близость пропозиций;

c) дополнительной лексической информации.

Эти условия удовлетворяются при использовании метрики редакционного расстояния, которое отражает число операций, необходимых для приведения строк к общему виду. Для различных прикладных задач были разработаны метрики Левенштейна («edit distance»: операции добавления, удаления, замены), Хемминга («Hamming distance»: операция замены), «Episode distance» (операция вставки), Дамерау-Левенштейна («Damerau–Levenshtein distance»: вставка, удаление или транспозиция) и прочие.
Стоимость операций может отличаться. Необходимо определить допустимые операции и определить их стоимости.
Доработаем метрику Дамерау-Левенштейна таким образом, чтобы стоимость операций зависела от операндов по принципу: стоимость операции снижается при наличии среди операндов стоп-слов или токенов, не представленных в грамматике G.
Пусть

- $del(prs, i)$ – оператор удаления токена с порядковым номером i из пропозиции prs;
- $ins(prs, i, tok)$ – оператор вставки токена tok в позицию i пропозиции prs, со сдвигом вправо токенов порядковых номеров $\overline{i..n}$;
- $trans(prs, i, j)$ – оператор транспонирования токенов с порядковыми номерами i и j в пропозиции prs.
- $sub(prs, i, tok)$ – оператор замены токена с порядковым номером i на токен tok в пропозиции prs.

Обозначим как $Cost(op)$ функцию стоимости операции. Введем обозначения:
$\mathrm{SE}_{\mathrm{notSt}} = prsTok_k \in (\varphi \cap \overline{StopWords})$,
$SE_{st} = prsTok_k \in (\varphi \cap StopWords)$,
$UW_{St} = prsTok_k \in (StopWords \setminus \varphi)$,
$UW = prsTok_k \notin (StopWords \cup \varphi)$, где
$prsTok_k$- токен с порядковым номером k в пропозиции $prsTok$.
Накладываемые ограничения:

1. $Cost(op) > 0 : op \in \{del, ins, trans\}$;

2. $Cost(del \mid ins) > Cost(trans)$;

   $Cost\big(del(\mathrm{SE_{notSt}})\big) \geq Cost\big(del(UW)\big) \geq Cost\big(del(\mathrm{SE_{st}})\big) \geq Cost\big(del(UW_{st})\big)$;
3. $Cost\big(ins(\mathrm{prs, i, SE_{notSt}})\big) \geq Cost\big(ins(\mathrm{prs, i}, UW)\big) \geq Cost\big(ins(\mathrm{prs, i, SE_{St}})\big) \geq$

   $Cost\big(ins(\mathrm{prs, i}, UW_{st})\big)$;
4. $Cost\big(trans(\mathrm{prs, SE_{notSt}, SE_{notSt}})\big) \geq Cost\big(trans(\mathrm{prs, SE_{notSt}}, UW)\big) =$

   $Cost\left(trans\big(\mathrm{prs, UW}, SE_{notStop}\big)\right) \geq Cost\left(trans\big(\mathrm{prs, UW, SE_{Stop}}\big)\right) =$

   $Cost\left(trans\big(\mathrm{prs, SE_{Stop}, UW}\big)\right) \geq Cost\left(trans\big(\mathrm{prs, SE_{Stop}}, UW_{st}\big)\right) =$

   $Cost\left(trans\big(\mathrm{prs}, UW_{st}, \mathrm{SE_{Stop}}\big)\right)$;
5. $Cost\left(sub\big(\mathrm{prs, SE_{notStop}, i}\big)\right) \geq Cost\big(sub(\mathrm{prs}, UW, \mathrm{i})\big) \geq$

   $Cost\left(sub\big(\mathrm{prs, SE_{Stop}, i}\big)\right) \geq Cost\big(sub(\mathrm{prs}, UW_{st}, \mathrm{i})\big)$;

2. Построение недетерминированного конечного автомата

Рассмотрим суффиксное дерево произвольной строки.

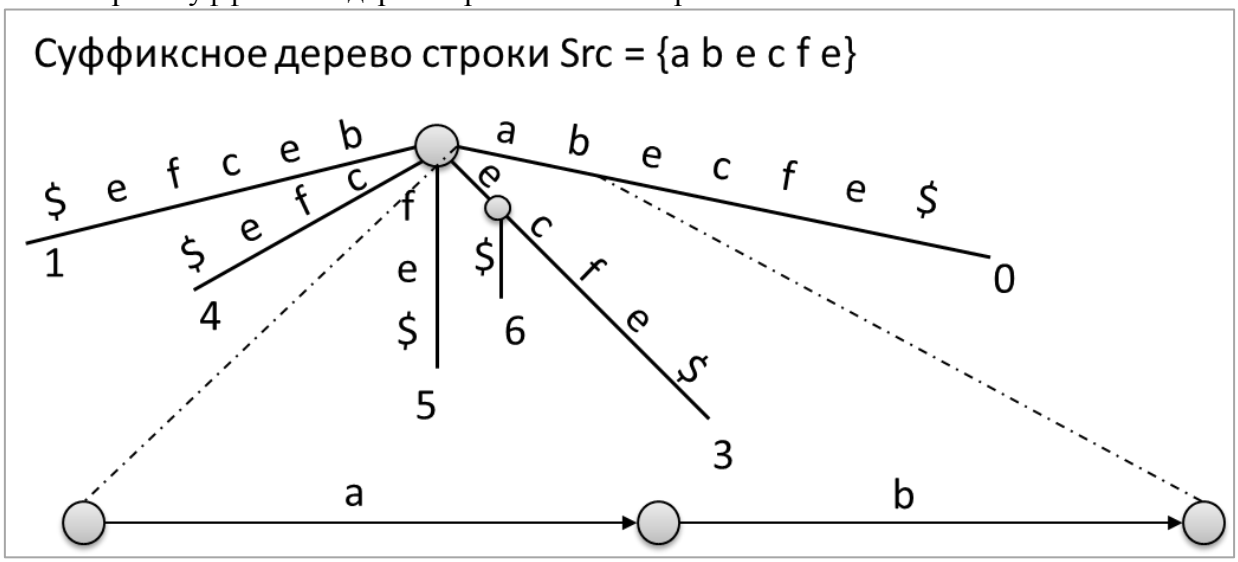

Рисунок 2. Суффиксное дерево.

Недетерминированный конечный автомат определения неточного вхождения подстрок с вычислением редакционного расстояния Дамерау-Левенштейна (далее НКА) должен строиться на основе полного суффиксного дерева.

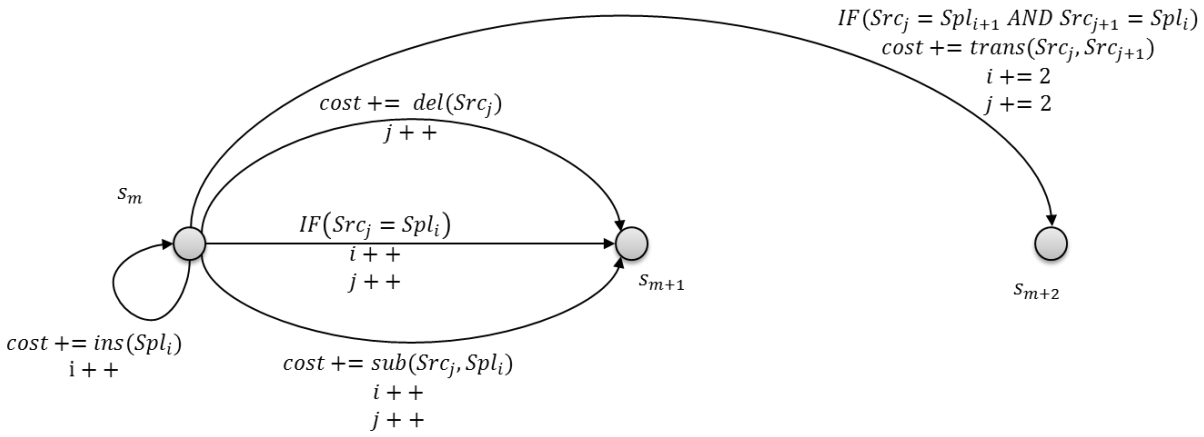

Рисунок 2. Фрагмент НКА для префикса "a".

Состояния НКА привязаны к узлам суффиксного дерева. Смена состояний НКА означает перемещение алгоритма по суффиксному дереву. Функция $cost$ является целевой функцией работы НКА. При превышении заданного пользователем порога производится переход в конечное состояние с отрицательной рекомендацией применения правила. Фрагмент не отображает начальное состояние и выход в завершающее, он иллюстрирует выходы из некоторого состояния s_m согласно возможным редакционным предписаниям Дамерау-Левенштейна.
Переход в следующее состояние s_{m+1} производится:

1. Без увеличения целевой функции при совпадении текущего символа в источнике и образце;
2. С увеличением целевой функции через применение операций del, sub на источнике (здесь и далее операции имитируются изменением индексов текущих символов, без преобразования суффиксного дерева и НКА).

Переход в состояние s_{m+2} связан с операцией транспонирования соседних элементов источника. Операция вставки (ins) позволяет обработать текущий символ без изменения состояния.
Суффиксное дерево предполагает возможность разветвления суффиксов, после префикса “e” иллюстрации (Рисунок 2). В этом случае выходы состояния дублируются для каждой исходящей ветви суффиксного дерева, отличаясь условиями переходов. Индекс j в этом случае будет индивидуален для каждой ветви.

3. Алгоритм моделирования с помощью событийно-статистической модели.

В построенной на предыдущих этапах событийно-статистической модели имеется весовая характеристика правил грамматики. Предлагается сделать эти свойства правил управляющими процессом поиска. В случае контекстно-зависимых правил стоимость применения правила определяется недетерминированным конечным автоматом (НКА). Алгоритм моделирования является детерминированным, поскольку вероятностный компонент содержится в исходных данных этапа построения модели, а последовательность операций моделирования строго определена для каждого набора входных данных. Из соображений ресурсных ограничений накладывается ограничение на

число операций вывода в грамматике. Постановка задачи исследования декларирует два типа запросов моделирования:

- Безальтернативный режим моделирования;
- Режим моделирования с альтернативой.

3.1. Безальтернативный режим моделирования

Исходными данными являются: **начальная пропозиция** (ЕЯ-описание) и **конечная пропозиция** (знак), в общем случае являющиеся не единичными. Поиск преобразований между пропозициями осуществляется через применение правил к пропозиции, имеющей наименьшую стоимость вывода. Особенностями задачи являются:

1. Сформированная в результате применения правил пропозиция может не полностью совпадать с искомой;
2. Требуется найти последовательности вывода в грамматике между исходной и целевой пропозицией с минимальной стоимостью.

Используем псевдоязык для описания алгоритма поиска:

```
алг Поиск вывода между пропозициями (
                                арг таб St[1:N] ,       // символы начальной пропозиции
                                арг таб Rt[1:M],        // символы конечной пропозиции
                                арг цел Max,            // максимальная длина искомой последовательности
                                арг цел Min,            // минимальная длина искомой последовательности
                                арг лог Shortest        // условие поиска наиболее вероятного пути
                                арг таб Rls[1:T],       // множество правил грамматики
                                рез таб R[1:K][1:L])    // таблица найденных последовательностей
   дано | Min >=1, Max <=7
   надо | Записать в R наиболее вероятные последовательности преобразований, ведущие от St к Rt.
нач
   таб CurrProp; // таблица текущих необработанных пропозиций
   Добавить в CurrProp St;
   лог fin = ложь; // признак завершения поиска
   нц пока CurrProp не пуст и fin = ложь
         таб maxP = элемент CurrProp с наименьшей стоимостью вывода;
         цел level = удаленность maxP от St в шагах преобразований;
         таб Tr[1:P] = найти в Rls все правила, применимые к maxP;
         нц для i от 1 до P
               trans = Tr[i];
```

1.4. Режим моделирования с альтернативой.

Исходными данными являются:

$$Query = (Sense,\ DerivOpt), \qquad (1)$$

где $Sense$ – естественно-языковое описание запроса;
$DerivOpt$ – параметры вывода в GExt и отбора результатов.

$$DerivOpt\ = (EdCosts, StepsLimit, GenLimit, AFilt), \qquad (2)$$

где $EdCosts$ – стоимость редакционных операций в соответствии с методикой интеграции когнем;
$StepsLimit, GenLimit$ – ограничения на число шагов вывода от исходной пропозиции и число произведенных грамматикой пропозиций;
$AFilt$ – множество допустимых когнитивных областей;
В поиске используется расширенная грамматика, представленная при построении грамматики. Обработка пропозиции производится аналогичной при построении грамматики. В процессе работы алгоритмом создаются новые пропозиции, которые размещаются в упорядоченной очереди RawSent. Выбор следующей пропозиции согласно минимальной стоимости вывода в грамматике для достижения пропозиции позволяет управлять последовательностью вывода с учетом ресурсных ограничений. При использовании потокобезопасной реализации очереди RawSent итерации внутреннего цикла могут выполняться параллельно, в рамках, например, многоядерной вычислительной машины.

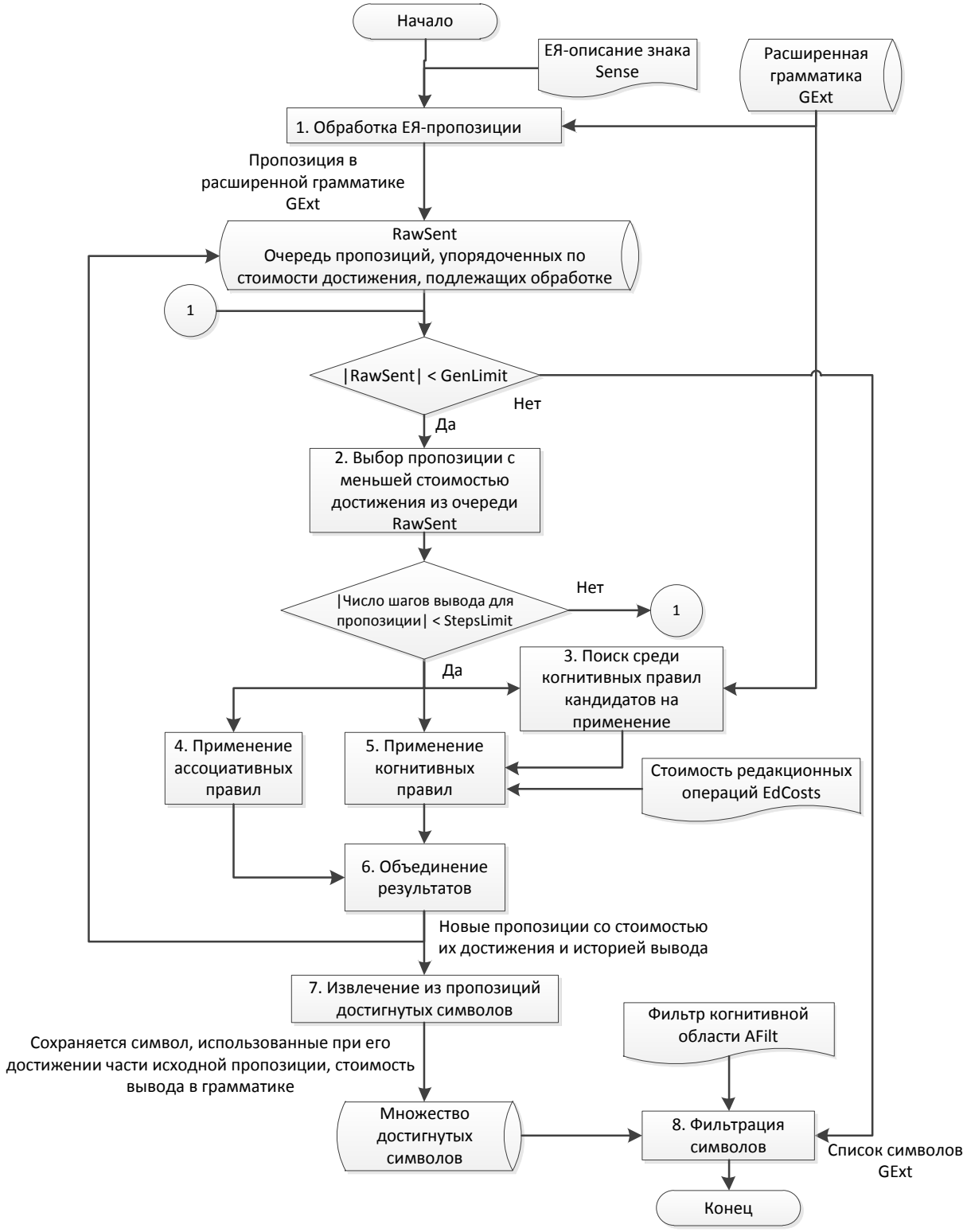

Рисунок 3. Алгоритм моделирования с альтернативой

При извлечении из пропозиции достигнутых символов игнорируются символы, присутствующие в ЕЯ-описании искомого знака (запросе).
Модель имеет ряд параметров, которые варьирует исследователь. Эти параметры касаются:

- Исходных данных для построения грамматики GExt;
- Параметров вывода в грамматике;
- Настроек НКА выполнения КЗ-правил при нечетком совпадении;

При подготовленной заранее модели набор операций, выполняемых исследователем, включает:

- Чтение из базы данных программного комплекса когнем;
- Фильтрация когнем и разделение результатов фильтрации на обучающую и тестовую выборки;
- Построение классификатора на основе обучающей выборки;
- Оценка классификатора с помощью выборки тестирования.

Для того, чтобы когнема могла быть использована для вывода в грамматике, построения и оценки классификатора, формула смысла и знак должны быть приведены к символам грамматики. Фильтрация когнем выполняется по: области когнемы, способу, диапазону длин пропозиции, проценту от общего количества.
Фильтры можно комбинировать в цепочку фильтров, определяя выборки на подобие **<выбрать половину когнем, имеющих самую популярную область, способ = "Дефиниция", а также длину формулы смысла от 1 до 5>.**
Разделение на обучающую и тестовую выборки производится в пропорции 1:1.
Для оценки модели выполним запросы для следующих выборок исходных данных:

1. 10 наиболее популярных областей:
2. {язык, быт, литература, зоология, спорт, ботаника, взаимоотношения людей, армия, природа, музыка};
3. 10 наиболее популярных способов:
4. {дескрипция, дефиниция, перифраза, суждение, метафора, синоним, множество, прец. текст, фрейм, импликация};
5. Пропозиции длинной от 1 до 5, 20% от общего числа;
6. Пропозиции длинной от 5 до 10, 20% от общего числа;

Для каждой выборки моделирование и оценка производятся независимо. Моделирование выполняется в в режиме использования информации об области когнемы-запроса с ограничением числа возможных знаков, либо без использования, что приближает эксперимент к типовому использованию вопросно-ответной системы.
Результаты моделирования формируют сводную таблицу достигнутых в процессе вывода символов грамматики (ЗНАКов) (Таблица 1).

В таблице приведена сводная таблица результатов моделирования, в которой для запросов указаны возможные языковые единицы с показателями Достижимость и Использование ФС.
Для оценки эти показатели переводятся в форму рангов по убыванию абсолютных значений. Так как моделирование производится для ЕЗМ, знаки которых известны, среди множества альтернатив могут быть помечены достоверно релевантные. Из когнем формируем 2 равные группы – обучающую выборку, на основе которой определяются коэффициенты линейного классификатора и выборку для тестирования.

Таблица 1. Результаты моделирования.

Знак	Достижимость(cost)		Использование ФС (usage)	
	значение	ранг	значение	ранг
{<Клен> <Раскудрявый житель леса, весь резной листвой одетый> <Метафора> <Ботаника> <Рецепт>}				
крона	3,528	1	7 / 7	1
дуб	1,379	2	7 / 7	1
кедр	1,11	3	7 / 7	1
тополь	0,839	4	7 / 7	1
трава	0,76	5	7 / 7	1
ствол	0,725	6	7 / 7	1
шелковица	0,697	7	7 / 7	1
клен	0,509	8	7 / 7	1
лист	0,497	9	7 / 7	1
{<Спорт> <Физические упражнения, направленные на достижение высоких результатов в соревнованиях> <Дескрипция> <Спорт> <Рецепт>}				
кросс	11,468	1	9 / 9	1
стол	8,54	2	9 / 9	1
спорт	5,694	3	9 / 9	1
бег	2,603	4	9 / 9	1
рост	0,975	5	9 / 9	1
{<Тротуар> <Пешеходная дорожка вдоль трассы> <Дефиниция> <Город> <Рецепт>}				
метро	0,291	1	4 / 4	1
зебра	0,188	2	4 / 4	1
тротуар	0,099	3	4 / 4	1
смог	0,008	4	2 / 4	3
трава	0,005	5	3 / 4	2

Литература

Navarro, 2001 Navarro G. A Guided Tour to Approximate String Matching. // ACM Computing Surveys (CSUR), 2001. № 33. pp. 31–88

Гасфилд, 2003 Гасфилд Д. Строки, деревья и последовательности в алгоритмах: Информатика и вычислительная биология. СПб.: Невский диалект; БХВ-Петербург, 2003. 654 с

Кормен и др., 2005 Кормен Т.Х. и др. Алгоритмы: построение и анализ. Пер. с англ. – М. : Издательский дом “Вильямс”, 2005. Вып. 2-е. 1296 с