УДК 025.4.036; 004.822

АЛГОРИТМЫ ВАЛИДАЦИИ ОТВЕТОВ В ЗАДАЧЕ ВОПРОСНО-ОТВЕТНОГО ПОИСКА

А. А. Соловьёв

МГТУ им. Н. Э. Баумана

Поступила в редакцию 20.10.2011 г.

Аннотация: Рассмотрена задача проверки ответов в вопросно-ответном поиске. Анализ литературы показал, что некоторые алгоритмы обработки семантических структур применимы к синтаксическим структурам, и наоборот. Обсуждается потенциальная практическая значимость таких комбинаций. Раработан план их экспериментального испытания на основе таблиц релевантности РОМИП, полученных после участия в 2010.
Ключевые слова: информационный поиск, вопросно-ответный поиск, вопросно-ответные системы, проверка ответов, валидация ответов, компьютерная лингвистика, обработка естественного языка.
Annotation. This paper describes Answer Validation Task in Question Answering. Literature study concluded with notice about practical applicability of some algorithms to syntactic structures despite they originally were applied to semantic, and vice versa. Running of missing experiments is planned to base on relevance tables, derived after participation in ROMIP seminar last year.
Key words: information retrieval, question answering, answer validation, natural language processing.

1. ВВЕДЕНИЕ

Вопросно-ответный поиск – это особый вид задачи информационного поиска, активно использующий методы компьютерной лингвистики. В отличие от классического поиска по ключевым словам, результатом поиска является не документ, а краткий и лаконичный фрагмент текста. Ответ ищется в коллекции документов, например в Интернет.

Наиболее успешно решается задача ответа на вопросы об определениях (*англ.: definitional*) и фактографические (*англ.: factoid*). Сегодня системы ограничиваются поиском ответа, явно встречающегося в тексте, и не занимаются логическим выводом новой информации.

Типичной архитектурой вопросно-ответной системы является архитектура метапоисковой системы, т.е. система надстраивается поверх классической системы поиска по ключевым словам (Рис. 1). Выделяют 4 подзадачи: анализ вопроса (*A1*), поиск фрагментов текста (*A2*), выделение ответов-кандидатов (*A3*) и проверка ответов (*A4*).

В работе [1] подробно обсуждается вероятностный подход для решения задач *A1, A2, A3*, а в [12] рассмотрена реализация модуля анализа вопроса, используемого здесь. В последующих разделах обсуждается только последняя подзадача типового конвейера – проверка ответа (*A4*). В [11] опубликован отчёт об участия в РОМИП 2010 системы Умба, разработанной автором.

В таблице ниже перечислены несколько вопросов из заданий РОМИП.

Во втором разделе обсуждается подзадача проверки ответа. В третьем разделе рассмотрены алгоритмы проверки ответов. В четвёртом разделе обсуждается возможность использовать синтаксическое и поверхностно-семантическое представление текста в каждом из рассмотренных алгоритмов.

2. ЗАДАЧА ПРОВЕРКИ ОТВЕТА

При оценке вопросно-ответной системы возникает серьёзная проблема: невозможность использовать полученные ранее таблицы релевантности в новых экспериментах. Результатом каждого задания является не просто краткий ответ, но и фрагмент текста из конкретного документа, явно подтверждающий этот ответ. Таких фрагментов может быть в коллекции много, и система может сделать вывод на основании какого-то одного из них или даже в условии избыточности – нескольких разных фрагментов в разных документах, содержащих один

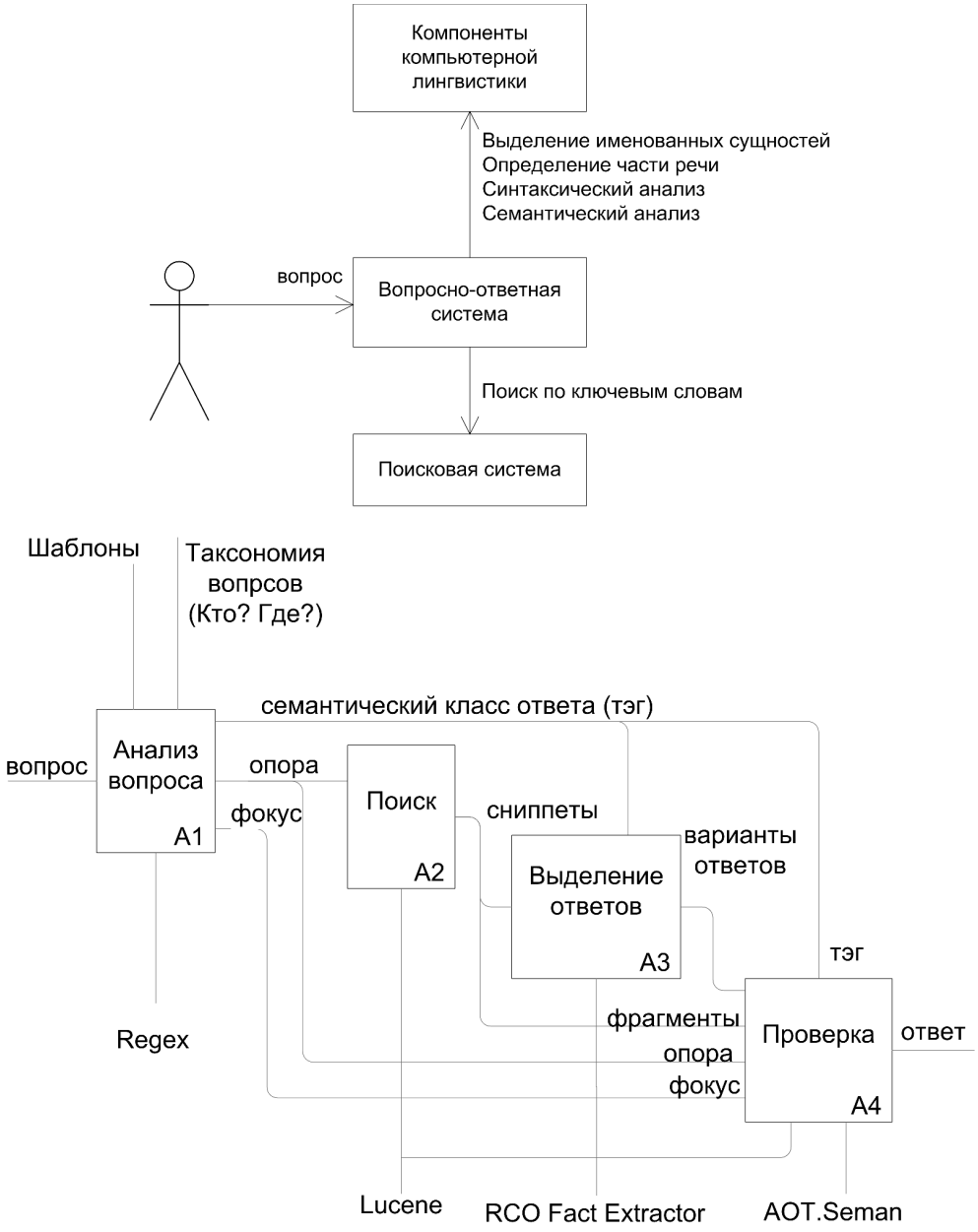

Рис. 1 Архитектура метапоисковой системы и подзадачи вопросно-ответного поиска

Таблица 1

Примеры заданий РОМИП (орфография оригинальная)

№	Вопрос, жирным шрифтом выделен фокус
nqa2009_6368	как отключить перехват клавиатуры?
nqa2009_7185	сколько стоит поченить гнездо у телефона сони эрикссон?
nqa2009_6425	в каких религиях как рассматривается карма?
nqa2009_3123	отечественная война кто с кем ?
nqa2009_8557	являются ли чердаки пожароопасными помещениями?
nqa2009_7801	какое колличество циклов чтения/записи предусмотренно компанией fujifilm для картриджей стандарта lto 4?

и тот же ответ. При этом в таблице релевантности, в отличие от классического поиска, окажется не только идентификатор документа, но и фрагмент текста (сниппет) и краткий ответ из этого фрагмента. Эта запись подтверждена асессором. В следующий же раз, когда исследователь захочет измерить качество модифицированной системы (вне ежегодной кампании), он не будет иметь доступа к тем же асессорам, но будет иметь таблицу релевантности прошлого года. Однако модифицированная система может найти новый фрагмент нового документа с тем же или даже новым вариантом ответа, который не встречался в предыдущих результатах. И это не означает, что система ошиблась. Подобная ситуация (новый неоценённый ранее документ) возможна и в классическом поиске, однако в случае вопросно-ответного поиска она гораздо более вероятна.

Подзадача проверки вопроса лишена этой проблемы. Модуль проверки должен для каждого кортежа *<вопрос, документ, фрагмент, ответ>* принять решение: да или нет. В такой формулировке таблица релевантности с позитивными и негативными примерами может быть успешно использована. Примером такого подхода к оценке является семинар CLEF Answer Validation Exercise [5]. Для оценки методов проверки вопросов мы будем использовать таблицу релевантности, построенную на основе результатов вопросно-ответной дорожки РОМИП 2010.

3. МЕТОДЫ ПРОВЕРКИ ОТВЕТОВ

Рассмотрим существующие алгоритмы проверки вопроса, основанные на вычисления схожести структуры вопроса и сниппета.

3.1. ПЕРЕСЕЧЕНИЕ МНОЖЕСТВ СЛОВ И/ИЛИ ГРАММАТИЧЕСКИХ ОТНОШЕНИЙ

Успешная в традиционном поиске модель мешка слов (*англ: bag of words*) часто применяется в базовом (*англ: baseline*) прогоне системы. Пусть Q – множество слов в вопросе, а T – множество слов во фрагменте подтверждающего текста. Тогда отношение $E = |Q \cap T|/|Q|$ может является мерой «подтверждения» ответа на вопрос Q.

Естественным усложнением является использование не множества слов, а множества синтаксических отношений, т.е. пар слов, связанных грамматической связью: *R(N1,N2)*. Пусть Q – множество таких кортежей-отношений в вопросе, а A – множество кортежей-отношений в ответе. Воспользовавшись той же формулой получаем $E = |Q \cap T|/|Q|$. В работе [7] этот метод используется в качестве «запасной стратегии» – в случае, когда более сложный алгоритм применить не удаётся по тем или иным причинам.

3.2. СОПОСТАВЛЕНИЕ СКАЗУЕМЫХ

В работе [6] используется усложнённая модификация формулы из предыдущего раздела – *Predicate Matching*. Ответ и сниппет предварительно проходят аннотацию семантическими ролями (*англ: Semantic Role Labeling*). В результате слова предложения получают метку либо сказуемого, либо аргумента при каком-то сказуемом. Сравниваются два сказуемых со всеми зависимыми словами: одно сказуемое из вопроса, другое – из сниппета. Схожесть двух сказуемых вычисляется как произведение лексической схожести глаголов (например, расстояние по словарю WordNet[2]) и схожести наборов аргументов:

$$Sim_{Pr \in d} = Sim_{V \in rb} \times Sim_{Args}$$

Схожесть аргументов предиката вопроса p_q и предиката p_a из сниппета вычисляется следующим образом:

$$Sim_{Args}\left(p_a, p_q\right) := \\ = \frac{\sum_{t_a \in T_a} \max_{t_q \in T_q} \left(Sim_{ExpTerm}\left(t_a, t_q\right)\right)}{\left|T_q\right| + \left|\left\{t_a \in T_a \middle| \max_{t_q \in T_q} \left(Sim_{ExpTerm}\left(t_a, t_q\right)\right) = 0\right\}\right|}$$

Последняя формула является модификацией формулы из 3.1, отметим их отличия: а) рассматриваются только аргументы при выбранных сказуемых; б) пара слов из вопроса и сниппета даёт вклад в меру схожести равную схожести этих слов; в) в случае если онтология WordNet не доступна (как для русского языка), то $Sim_{ExpTerm}$ принимает только два значения: 0 или 1.

Если в вопросе или сниппете несколько сказуемых – то вычисляется схожесть всех пар. Наибольшее из полученных чисел и будет серой подтверждения ответа сниппетом.

3.3. РАССТОЯНИЕ РЕДАКТИРОВАНИЯ ДЛЯ ДЕРЕВЬЕВ

В работе [4] рассматривается задача вычисления схожести деревьев грамматических зависимостей между словами двух предложений:

вопросительного и повествовательного. В отличие от формул выше, деревья сравниваются в целом, а не в контексте отдельных отношений/предикатов. Авторы [4] применили естественную метрику схожести деревьев: минимальное число операций редактирования, необходимых для трансформации одного графа в другой. Доступные операции редактирования: удаление вершины, вставка, замена – представлены на Рис. 2.

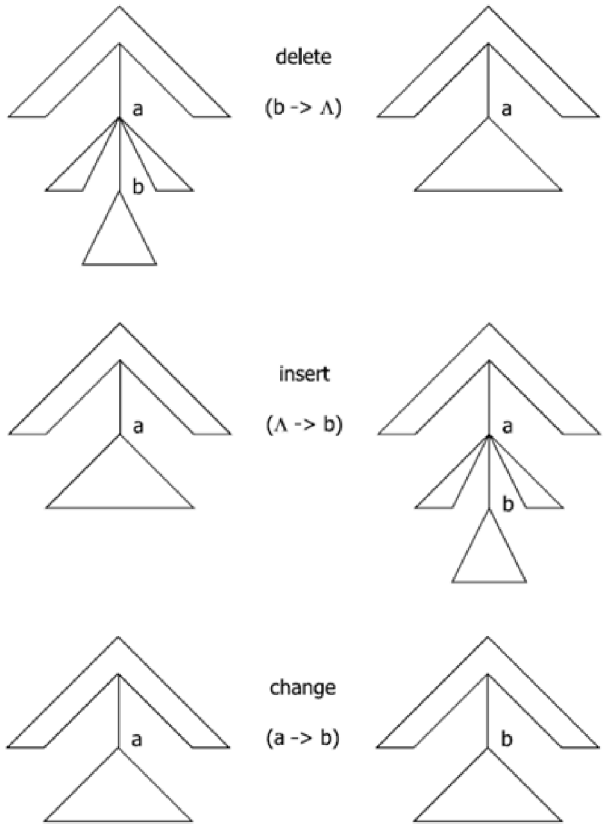

Рис. 2. Элементарные операции редактирования дерева

Разным операциям приписан разный вес: $\gamma(a \to \lambda)$, $\gamma(\lambda \to a)$, $\gamma(a \to b)$.

Пусть $S = \langle s_1; s_2; \ldots; s_k \rangle$ – последовательность операций, приводящая к трансформации дерева сниппета в дерево вопроса. Тогда стоимость этой трансформации есть сумма стоимостей операций: $\gamma(S) = \Sigma\, \gamma(s_i)$. Требуется найти самую дешёвую последовательность операций редактирования:

$$\delta(T_p, T_q) = \\ = \min_S \left\{ \gamma(S) \mid S(T_p) = T_q \right\}$$

Также алгоритм поиска предписания редактирования модифицирован таким образом, чтобы удаление лишних поддеревьев в подтверждающем тексте не штрафовалось, т.к. текст с ответом почти всегда содержит дополнительные грамматические конструкции, не относящиеся к вопросу.

$$DR(T_q, T_a) = \min_{F \in F(T_a)} \delta(T_q, T_{a \setminus F})$$

$$DR(T_q, T_{a \setminus F}) \min_{S\{\gamma(S) | S(T_q) = T_{a \setminus F}\}}$$

Где $F(T)$ – множество всех возможных поддеревьев, а S множество всех возможных последовательностей операций редактирования γ.

В работе [8] разработан эффективный алгоритм на основе динамического программирования для решения поставленной задачи. Серьёзным недостатком же является требование к упорядоченности деревьев: в случае естественного языка, допускающего разный порядок слов, более адекватной моделью представления текста являются непорядочные деревья.

3.4. СОПОСТАВЛЕНИЕ ДЕРЕВЬЕВ ЗАВИСИМОСТЕЙ

В работе [3] предлагается сравнивать два дерева зависимостей в задаче лексического вывода, используя алгоритмы, применяемые для обработки параллельных двуязычных текстов. Для двух деревьев D и D' строится матрица соответствия элементов M размера NxN', где N – число элементов в дереве D, а N' – число элементов в дереве D'. Каждый элемент $S(v,v')$ в матрице вычисляется по следующим рекурсивным формулам:

$$S(v, v') = \max(TreeMatch(v, v'), \\ \max_i S(v_i, v'), \max_j S(v, v'_j) - SP)$$

Мера схожести двух вершин равна либо мере схожести поддеревьев, вершинами которых они являются, либо схожести непосредственного потомка первой вершины v_i со второй вершиной v', либо схожести первой вершины v с непосредственным потомком второй v'_j, но за вычетом штрафа SP за пропуск вершины. Несимметричность штрафа за пропуск обусловлена намерением разрешить пропускать слова сниппета без штрафа, т.к. сниппет обычно содержит больше информации, чем требуется для ответа на вопрос. Это поведение алгоритма аналогично правилу удаления лишних поддеревьев в разделе 3.3.

Мера схожести двух поддеревьев, вершинами которых являются v и v', является линейная комбинация лексической схожести этих корневых вершин и некоторой мере схожести двух множеств непосредственных потомков:

$$TreeMatch(v, v') =$$
$$= PW \cdot ParentMath(v, v') +$$
$$= (1 - PW) \cdot ChildMath(v, v')$$

ParentMatch – есть некоторая лексическая схожесть двух слов, аналогичная той, что использовалась в разделе 3.2.

$$ChildMatch(v, v') =$$
$$= \max_{p \in P(v,v')} \left[\sum_{(i,j) \in p} \frac{|v'_j|}{|v'|} \cdot S(v_i, v'_j) \right]$$

Мера схожести двух множеств непосредственных потомков вершин v и v' вычисляется как стоимость наилучшего из всех возможных отношений между этими двумя потомками. Здесь $P(v,v')$ – множество всех подмножеств пар непосредственных потомков. Заметим, что последняя формула рекурсивно ссылается на первую. Эта рекурсия не будет бесконечной в силу ацикличности деревьев.

Элементы $S(v,v')$ составляют матрицу M. За меру схожести двух текстов принимается один единственный элемент этой матрицы – лучшее найденное соответствие корневой вершины гипотезы (обычно сказуемое-глагол).

3.5. ПАРАЛЛЕЛЬНЫЙ ОБХОД ГРАФОВ

В предыдущей работе [11] был предложен оригинальный метод неточного сравнения семантических графов параллельным обходом в глубину.

Рассмотрим пример семантического графа для вопроса nqa2009_2256 «*кто использовал стволовые клетки?*» и фрагмента из документа 419883 «*Ученые использовали мезенхимные стволовые клетки, извлеченные из образцов костного мозга мужчин-добровольцев.*» (Рис. 3). Графы построены библиотекой AOT. Seman [9].

В основе метода лежит интуиция, что если у простого вопроса «кто?» или «где?» заменить фокус вопроса (вопросительное слово) кратким ответом, мы получим семантически верное утверждение. Мы не рассматриваем проблему грамматической корректности полученного предложения. На Рис. 3 подграф УЧЕНЫЕ-ИСПОЛЬЗОВАЛИ-КЛЕТКИ-СТВОЛОВЫЕ во фрагменте очевидным образом соответствует графу вопроса КТО-ИСПОЛЬЗОВАЛИ-КЛЕТКИ-СТВОЛОВЫЕ, если заменить КТО на УЧЕНЫЕ. Любой строгий алгоритм поиска изоморфизма подграфов обнаружит это равенство подграфов.

Однако, более часты случаи с менее строгим совпадением подграфов. Например, вопрос nqa2009_856: «*где собирают меганы?*» и фрагмент из документа 477114: «*Может это от части потому, что часть Сценников, как и Меганов, собиралась в Турции*».

Здесь присутствуют узлы-связки однородных членов. Чтобы обойти эту проблему предлагается простая предварительная обработка графа сниппетов – добавить в граф дуги, огибающие каждую вершину:

• Для каждой вершины v, для каждой пары (e_i, e_o) её входящих и исходящих дуг, создать новую дугу $(src(e_i), trg(e_o))$, где $src(x)$ – вершина, откуда выходит дуга x, $trg(x)$ – вершина, в которую входит дуга x. Пометить эту дугу меткой **shortcut**.

• Для каждой вершины v, которая имеет две исходящие дуги e_{o1}, e_{o2}, но ни одной входящей (например, в случае сказуемого), создать две новых дуги: $(trg(e_{o1}),trg(e_{o2}))$, $(trg(e_{o2}),trg(e_{o1}))$. Пометить меткой **shortcut**.

Алгоритм вычисления меры схожести подграфов выглядит следующим образом:

1. Найти вершину v_a с фокусом в вопросе, и v_q с ответом в сниппете.

2. Создать множество посещённых вершин $visited = \varnothing$.

3. Выполнять рекурсивно следующую функцию: $Visit(v_q, v_a, visited)$:

3.1. Если хотя бы одна из этих вершин v_q, v_a уже присутствует во множестве *visited*, то выходим из процедуры *Visit*, вернув результат *0*. Иначе добавляем v_q и v_a во множество *visited* и переходим на следующий шаг.

3.2. Инициализируем переменную-аккумулятор баллов: *result:=0*.

3.3. Для каждой дуги $e_{0q} \in out(v_q)$, где $out(x)$ – множество исходящих из вершины x дуг:

3.3.1. Для каждой дуги $e_{0a} \in out(v_a)$

3.3.1.1. Если семантическая схожесть слов $sim(trg(e_{0q}),trg(e_{0a}))>0$, то вычисляем рекурсивно функцию $s:= Visit(trg(e_{0q}), trg(e_{0a}))$.

3.3.1.2. Если $label(trg(e_{0a}))$=**shortcut**, то $s:=s*0.5$.

3.3.2. Из всех значений s, вычисленных на шаге 3.3.1.2, находим максимальное, т.е. балл, соответствующий наилучшему мходству дуги e_{0q} с некоторой дугой e_{0a}.

3.3.3. result:= result +max(s)

3.4. Аналогично для каждой исходящей дуги $e_{iq} \in inc(v_q)$, где $inc(x)$ – множество исходящих из вершины x дуг:

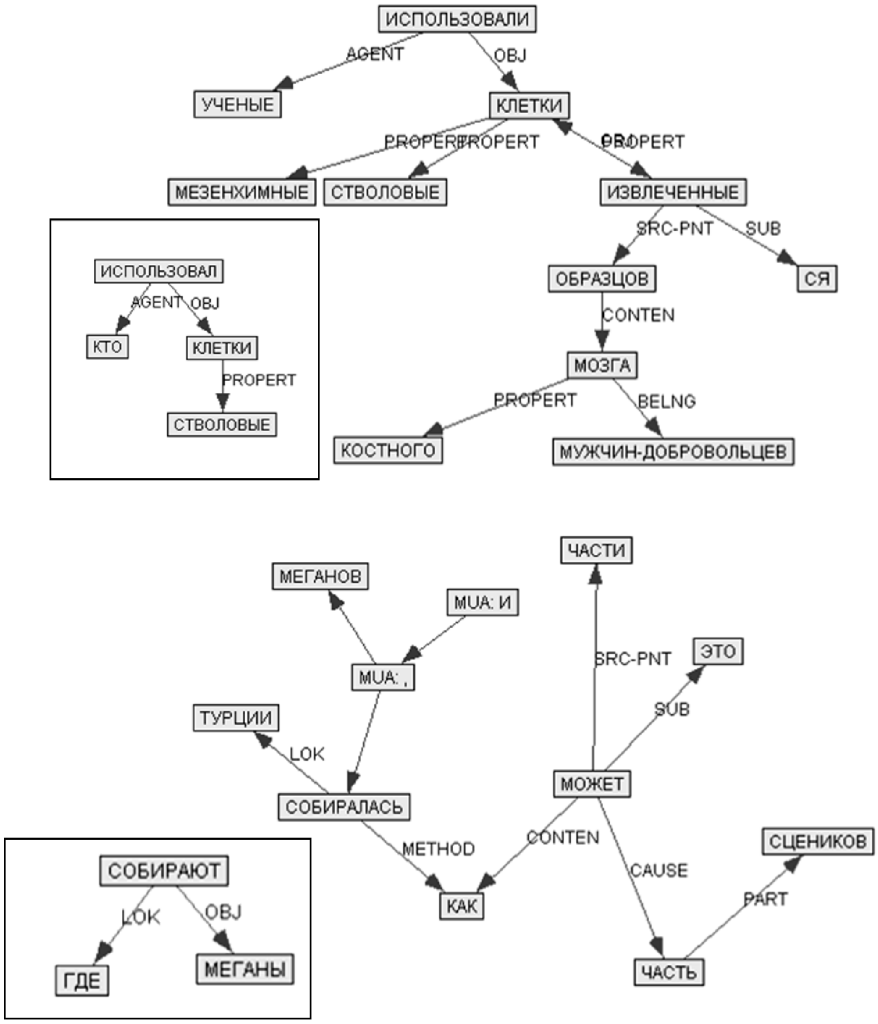

Рис. 3. Семантические графы для вопросов «кто использовал стволовые клетки?», «где собирают меганы?», и соответствующих сниппетов с ответом

3.4.1. Для каждой дуги $e_{ia} \in inc(v_a)$

3.4.1.1. Если $sim(trg(e_{iq}), trg(e_{ia}))>0$, то вычисляем рекурсивно функцию $s:= Visit(trg(e_{iq}), trg(e_{ia}))$.

3.4.1.2. Если $label(trg(e_{ia}))=*shortcut*$, то $s:=s*0.5$.

3.4.2. Из всех значений s, вычисленных на шаге 3.4.1.2 находим максимальное.

3.4.3. result:= result +max(s)

4. Полученное число *result*, вычисленное для исходной пары затравочных вершин, соответствующее фокусу вопроса и ответу, нормируем на число вершин в графе вопроса: *result: = result / |Vq|*.

Отметим, что семантическое схожесть слов в вершинах графа вычисляется по лексической онтологии WordNet, например *Lin Similarity[2]*. Однако для русского языка мы используем выработанное правило: если леммы совпадают, то 1, иначе 0.

Следующие рекурсивные формулы соответствуют описанному алгоритму:

$$s(v_q, v_a) = \begin{cases} 1 + sim_{inc}(v_q, v_a) + sim_{out}(v_q, v_a), \\ if\ sim(v_q, v_a) > 0 \\ 0, otherwise \end{cases}$$

$$sim_{inc}(v_q, v_a) = \sum_{e_q \in inc(v_q)} \max_{e_p \in inc(v_a)} s(src(e_q), src(e_a))$$

$$sim_{out}(v_q, v_a) = \sum_{e_q \in out(v_q)} \max_{e_p \in out(v_a)} s(trg(e_q), trg(e_a))$$

В отличие алгоритмов, рассмотренных в разделах 3.3 и 3.4, алгоритм сразу начинает

работу от известных ему пар сопоставленных вершин – от слова-ответа в подтверждающем тексте и от вопросительного слова в вопросе. Алгоритм сопоставления предикатов же вынужден рассмотреть все пары предикатов (глаголов), а алгоритм сопоставления вершин вообще перебирает все возможные пары вершин. Не стоит рассматривать данное свойство как способ экономии процессорного времени. Алгоритмы 3.3 и 3.4 были заимствованы из другой прикладной задачи – машинного перевода – когда как предложенный алгоритм с самого начала использовал специфику простых фактографических вопросов и ответов: пара вершин для старта алгоритмов сравнения уже известна, её не надо искать.

4. ПОДМЕНА МОДЕЛИ ДЛЯ НЕКОТОРЫХ АЛГОРИТМОВ

Рассмотренные выше алгоритмы в оригинальных работах использовались на одной из моделей: синтаксической (грамматика зависимостей) или семантической. В этой работе предлагается рассмотреть возможность применения алгоритма на другой модели: подменить семантические отношения синтаксическими отношениями, и наоборот. В таблице ниже приведено соответствие алгоритмов и моделей, найденное в литературе. Пустые позиции A, B, C, D, E являются областью интереса в данной работе.

Вот некоторые общие для всех экспериментов шаги:

1. Подготовить коллекцию из нескольких десятков русскоязычных кортежей <*вопрос, фрагмент, ответ, да/нет*>. Вопросы взять из заданий РОМИП (вопросы что/где), фрагменты из результатов РОМИП и из выдачи Яндекса.

2. Дерево синтаксических зависимостей строить в два этапа:

2.1. Синтаксический разбор на основе грамматики составляющих.

2.2. Построение дерева зависимостей на основе полученного разбора на составляющие (см. Досемантический анализ в [10]).

3. Граф семантических отношений строить с помощью AOT.Seman.

4. Результаты работы оценивать с помощью метрики «ошибка» – отношение числа неправильно принятых решений к общему числу решений.

5. Заключение

Рассмотрена подзадача проверки ответов в вопросно-ответном поиске. Обзор литературы выявил, что есть практическая возможность применить алгоритмы, оригинально реализованные для семантических структур, к синтаксическим структурам, и наоборот. Следовательно, необходимо экспериментально оценить таких комбинаций в области поверхностно-семантического анализа в качество решения задачи проверки ответа. Разработан план, состоящий из 11 11 экспериментов на вопросах «кто? где?» и таблиц релевантности РОМИП'2010. Пять из них – воспроизведение экспериментов других авторов, но для русского языка, один – повторение нашего эксперимента РОМИП 2010 [11]. Остальные 5 экспериментов будут поставлены впервые.

СПИСОК ЛИТЕРАТУРЫ

1. *Ittycheriah, Abraham.* A Statistical Approach for Open Domain Question Answering // Advances in Open Domain Question Answering. Springer Netherlands. Part 1. Vol. 32. – 2006.

2. *Lin, Dekang.* An Information-Theoretic Definition of Similarity // Proceedings of the 15th ICML, San Francisco, USA, 1998.

3. *Marsi, E., Krahmer, E., Bosma, W.E. and Theune, M.* Normalized Alignment of Dependency Trees for Detecting Textual Entailment. // Second

Таблица 2

Соответствие алгоритмов и моделей

Модели текста / Алгоритмы	Мешок слов	Грамматические зависимости	Поверхностно-семантические отношения
Пересечение множеств	[7]	A	[7]
Сопоставление предикатов		B	[6]
Сопоставление вершин		[3]	C
Расстояние редактирования		[4]	D
Параллельный обход		E	[11]

PASCAL Recognising Textual Entailment Challenge, Venice, Italy. – 2006.

4. *Panyakanok, V., Roth, D. and Yih, W.* Natural language interface via dependency tree mapping: An application to question answering // AI and Math. – January 2004.

5. *Rodrigo, Á., Peñas, A., and Verdejo, F.* Overview of the answer validation exercise 2008. In Proceedings of the 9th CLEF // Lecture Notes In Computer Science. Springer-Verlag, Berlin, Heidelberg. – 2009.

6. *Schlaefer, Nico.* A Semantic Approach to Question Answering. – 2007.

7. *Wang, R. and Neumann, G.* Using Recognizing Textual Entailment as a Core Engine for Answer Validation // Working Notes for the CLEF 2008 Workshop.

8. *Zhang, K. and Shasha, D.* Simple fast algorithms for the editing distance between tree and related problems // SIAM J. COMPUT. Vol. 18, No. 6. – 1989.

9. Автоматическая Обработка Текста [Электронный ресурс]. URL: http://aot.ru.

10. *Сокирко А. В.* Семантические словари в автоматической обработке текста // Диссертация на соискание ученой степени к.т.н.: М. 2001.

11. *Соловьев А.А.* Кто виноват и где собака зарыта? Метод валидации ответов на основе неточного сравнения семантических графов в вопросно-ответной системе. // Труды РОМИП 2010: Казань, 2010.

12. *Соловьёв А.А., Пескова О.В.* Построение вопросно-ответной системы для русского языка: модуль анализа вопросов // Новые информационные технологии в автоматизированных системах: материалы 13 научно-практического семинара. – МИЭМ. – 2010. URL: http://nps.itas.miem.edu.ru/2010/sbornik13.pdf.

Соловьёв А. А. – ООО «Аплана Международные проекты», инженер-конструктор. Московский государственный технический университет им. Н. Э. Баумана, аспирант. Тел. +79164884558. E-mail: a-soloviev@mail.ru

Solovyev Alexander A. – Engineer-Constructor at Aplana Software. PhD student at Bauman Moscow State Technical University. Tel. +79164884558. E-mail: a-soloviev@mail.ru