УДК 004.8
А.Е. Ермаков, к.т.н., ведущий специалист по компьютерной лингвистике, ermakov@rco.ru

ООО "ЭР СИ О", Москва

Язык семантических трансформаций для компьютерной интерпретации текста

Аннотация

Статья посвящена развитию подхода к семантической интерпретации текста, описанному автором в 2009 году. Подход основан на преобразовании результатов машинного синтаксического анализа предложения в сеть связанных фреймов, с последующей интерпретацией определенных конфигураций связей как новых фреймов, описывающих требуемые ситуации предметной области. Разработан формальный язык, позволяющий описать и нужным образом интерпретировать сколь угодно сложные конфигурации синтаксических связей между словами текста посредством описания и интерпретации отдельных пар связей между словами. Рассмотрены практические приложения к извлечению из текста описаний событий и фактов, а также к интерпретации оценочных высказываний об объектах.

Ключевые слова: компьютерный анализ текста на естественном языке, семантические представления, синтаксический анализ текста, семантическая интерпретация текста, сеть фреймов, трансформация пропозиций, извлечение событий и фактов, анализ оценочных высказываний.

Введение

Начиная с 1970-х годов, существуют и развиваются различные подходы к семантическому представлению и интерпретации значений высказываний на естественном языке (ЕЯ), используемые в системах компьютерного анализа текстов. К основным из зарубежных подходов можно отнести теорию представления дискурсов, теорию концептуальных графов, эпизодическую логику [1], фреймово-семантический парсинг [2], подход на основе нотации абстрактного представления смысла [3]. Обстоятельным, но уже неполным введением в современное состояние компьютерной обработки ЕЯ является справочник [4].

Пионером в области формализации семантики ЕЯ в конце 1960-годов стал американский философ и специалист в области математической логики Р. Монтегю. Согласно его идеям, значение ЕЯ-высказывания может быть представлено выражением на метаязыке - логической формулой, переменными которой являются значения элементов

высказывания – слов и синтаксических конструкций. Монтегю разработал и первый формальный метаязык для интерпретации значений английских предложений [5] в терминах логики предикатов. В нашей стране основоположниками разработки метаязыка на базе аппарата лексических функций в начале 1970-х выступили Ю.Д. Апресян [6] и И.А. Мельчук [7], решавшие задачу автоматического перевода. Современные достижения возглавляемого сегодня Апресяном коллектива воплощены в системе автоматического перевода ЭТАП-3 и ряде других приложений [8]. Например, при помощи аппарата лексических функций одно из значений слова *потушить* может быть описано формулой `Perf Caus(Им, Fin Lab(Вин,ОГОНЬ))`, где `ОГОНЬ` - это базисное, далее не разложимое понятие, а `Perf`, `Caus`, `Fin`, `Lab` - это базисные лексические функции, `Им` и `Вин` – обозначения падежей слов-аргументов функций. Так, базисная функция `Lab(x,y)` обозначает, что аргумент, обозначаемый словом *x*, подвергается действию аргумента, обозначаемого словом *y*. Значение предложения описывается математическим выражением. Например, предложению *Пожарные потушили загоревшийся склад* соответствует выражение `Perf Caus (ПОЖАРНЫЕ, Fin Lab(Perf Incep Labo1(СКЛАД, ОГОНЬ), ОГОНЬ))`, которое в обратном переводе на русский язык имеет следующее значение: *Пожарные сделали так, что перестал подвергаться действию огня начавший подвергаться действию огня склад*. Задача создания семантического метаязыка заключается в выборе системы таких базисных функций и понятий, позволяющих описать значения всех других понятий и предложений, дальнейшее толкование которых либо невозможно, либо нецелесообразно. Так, метаязык, разработанный под руководством В.А. Тузова и описанный в работе [9], содержит 72 базисные функции и около шестисот базисных понятий, на основе которых была сделана попытка описать значения всех слов русского языка с целью последующей семантической интерпретации любого русского предложения.

Современные обзоры формальных семантических описаний и их приложений можно найти в работах зарубежных авторов [10, 11]. Дополнительно следует отметить широту перспектив для компьютерной семантики ЕЯ, открываемых теорией К-представлений - концептуальных представлений, которая развивается в Москве с 1980-х годов В.А. Фомичевым. Первоначально теория К-представлений (ТКП) называлась теорией К-исчислений и К-языков. ТКП является центральной частью Интегральной формальной семантики ЕЯ - научного направления на стыке математической информатики, математической лингвистики и компьютерной лингвистики [14].

В статье [12] и монографиях [13, 14] предлагается оригинальная (по гинотезе автора - универсальная) математическая модель для описания структурированных значений предложений и связных ЕЯ-текстов, а также содержания посланий компьютерных

интеллектуальных агентов в многоагентных системах [15]. Модель представляет собой определение нового класса формальных языков, названных стандартными концептуальными языками (СК-языками). Проведено исследование выразительных возможностей СК-языков и показано, что выражения СК-языков удобно использовать для построения семантических описаний предложений, выражающих высказывания, вопросы, команды, распоряжения, советы, включая предложения со сложными описаниями множеств, однородными членами предложения, причастными оборотами, придаточными цели и придаточными определительными предложениями. Проведено сравнение выразительных возможностей СК-языков с выразительными возможностями других, наиболее часто используемых подходов к формальному представлению значений ЕЯ-текстов: теории представления дискурсов, теории концептуальных графов, эпизодической логики, теории расширенных семантических сетей, теории неоднородных семантических сетей и компьютерной семантики русского языка. Работы [13, 14] также описывают алгоритмы семантико-синтаксического анализа ЕЯ-текстов, предоставляемые ТКП. Работа [14] описывает применения разработанных алгоритмов к проектированию компьютерных систем обработки ЕЯ-текстов.

Для иллюстрации выразительных возможностей наиболее развитых современных языков описания семантики приведем ряд интуитивно понятных примеров описания ЕЯ-выражений средствами СК-языков из работы В.А. Фомичева [15]:

```
"Профессор Новиков прилетел вчера из Праги" => Ситуация(e1, прилет*(Время, x1)(Агент1, нек.чел.*(Квалиф, прафессор)(Фамилия, «Новиков») : x2)(Место1, нек.город*(Название, "Прага"): x3)) ^ Раньше (x1, Сейчас);

"3 контейнера с керамикой из Индии" => нек.множ *(Колич, 3) (Кач-состав, Контейнер1*(Содерж1, нек.множ*(Кач-состав, изделие*(Вид, керамика) (Страна, Индия))));

"Партия керамики, состоящая из коробок с номерами 3217, 3218, 3219" => нек.партия2*(Колич, 3)(Предм-состав, (нек.коробка1*(Номер, 3217) : x1 ^ нек.коробка1 *(Номер, 3218): x2 ^ нек.коробка1*(Номер, 3219) : x3));
```

Подобные метаязыки необходимы в компьютерных приложениях, требующих максимально подробной интерпретации смысла ЕЯ-выражений: в программах автоматического перевода, вопросно-ответных и экспертных системах (наиболее мощная из которых - IBM Watson: *https://ru.wikipedia.org/wiki/IBM_Watson*) и даже в системах общения компьютерных агентов, ставших предметом практического исследования в последнее десятилетие. В то же время на практике существует множество задач, для которых, с одной стороны, выразительная способность таких языков оказывается избыточной, а с другой

стороны, существуют значительные ограничения на человеческие ресурсы, которые могут быть затрачены для построения детальных семантических описаний значений всех слов и ЕЯ-конструкций в предметной области работы системы. К числу таковых относятся, например, задача извлечения из текста описаний заданных событий и фактов с их участниками, а также задача интерпретации высказываний об объектах как позитивных/негативных оценок этих объектов, известная также под названием “оценка тональности высказываний”. Решением двух указанных задач автор активно занимался на протяжении последних десяти лет в компаниях “ЭР СИ О” (*http://www.rco.ru*) и “Крибрум” (*http://www.kribrum.ru*), специализирующихся на компьютерной лингвистике и анализе Интернет-контента, практически воплощая описываемый в данной статье подход к семантической интерпретации текста на основе традиционных семантических представлений в форме сетей фреймов.

В рамках данного подхода разобранная в начале статьи фраза *Пожарные потушили загоревшийся склад* может быть семантически проинтерпретирована как пара фреймов, описывающих пару связанных фактов: `ПОТУШИТЬ { Субъект=ПОЖАРНЫЕ, Объект=СКЛАД } & ЗАГОРЕТЬСЯ { Субъект=СКЛАД }`. Далее, при наличии дополнительных правил интерпретации конструкций с глаголами *потушить* и *загореться* может быть сделан вывод, что в тексте описана негативная ситуация для объекта *склад* и позитивная для объекта *пожарные*. Если в тексте дополнительно указана информация, позволяющая идентифицировать данные объекты, например, *пожарные МЧС* и *склад ГСМ РЖД*, то фраза будет проинтерпретирована как позитивная оценка для МЧС и негативная для РЖД. Применение же трех правил интерпретации общеязыковых конструкций *приступить к* (*чему-то*), *завершение* (*чего-то*) и *операция по* (*чему-то*) позволит провести полностью аналогичную интерпретацию фразы *Пожарные приступили к завершению операции по тушению загоревшегося склада*, не вдаваясь в семантическое описание избыточных деталей.

Развиваемый автором подход, базовая реализация которого была разработана в компании “ЭР СИ О” и описана ранее в работе [16], основан на формальном описании подграфов в сети синтактико-семантических отношений между словами ЕЯ-выражения, соответствующих различным способам языкового описания искомой ситуации, с последующей интерпретацией определенных узлов подграфа как слотов фрейма в определенных ролях. Для каждого типа ситуаций создаются шаблоны - графы-образцы, которые ищутся как подграфы в сети синтактико-семантических отношений между словами, построенной синтаксическим анализатором, что позволяет абстрагироваться от особенностей поверхностно-синтаксической организации предложений, обеспечивая высокую инвариантность шаблонов к способам выражения ситуации в тексте. Существенными

недостатками метода являются: необходимость использования графических средств просмотра/редактирования шаблонов в форме графов, громоздкость и сложность просмотра описаний, вызванные тем, что одновременная работа с множеством графов-шаблонов, каковых для одной ситуации-фрейма бывает более десятка, затруднительна.

Излагаемое далее развитие подхода избавлено от указанных недостатков и описывает языковые способы выражения ситуаций в форме выражений на специальном языке, названном автором языком семантических трансформаций (ЯСТ).

Семантический интерпретатор и сети фреймов

В терминологии искусственного интеллекта фреймом называется схема ситуации или предмета. Фрейм имеет имя, которое идентифицирует класс описываемых им ситуаций или предметов, а также содержит слоты, которые имеют свои имена, идентифицирующие роли участников ситуации или ее характеристики, части или свойства предмета: `ПОКУПКА { Покупатель, Продавец, Товар, Цена|Стоимость }; АВТОМОБИЛЬ { Марка, МощностьДвигателя, МаксимальнаяСкорость... }`. Фрейм со слотами, заполненными текстовыми значениями, представляет конкретную ситуацию, описанную в тексте, и называется фреймом-экземпляром: `ПОКУПКА { Покупатель = КОРЕЙКО АЛЕКСАНДР, Продавец = БЕНДЕР ОСТАП, Товар = ПАПКА, Цена|Стоимость = МИЛЛИОН РУБЛЕЙ }`.

Семантический интерпретатор (СИ) работает с представлением текста каждого предложения в форме сети фреймов-экземпляров, в которой каждому слову предложения соответствует свой фрейм. Изначально имена фреймов представляют нормальные формы соответствующих слов. Фрейм имеет набор атрибутов - пар вида `ИмяАтрибута=ЗначениеАтрибута`, например, атрибуты с именем `semantic`, которые принимают значения обобщенной части речи (`semantic=_Noun`, `semantic=_Verb`...) и семантического разряда слова (`semantic=Event`, `semantic=Feature`, `semantic=Object:Animated`...). В формальном языке СИ имя фрейма представляет собой значение особого атрибута с именем `name`, при этом фрейм может иметь множество альтернативных имен: `name=покупать|покупка|купля-продажа`. Фрейм может представлять не только отдельное слово, но и многословную сущность, выступающую в языке как единое целое: `name=ООО "Рога и копыта", semantic=_Noun|Organization:Named`, или `name=9 мая 1945, semantic=_Adverb|Date`. Слоты фрейма могут быть заполнены ссылкой на другие фреймы, обычно представляющие синтаксически подчиненные слова. Строковым значениям слотов соответствуют имена связанных фреймов. Вхождение фрейма B во фрейм A в

качестве слота означает наличие направленной связи от фрейма А к фрейму В. Имена слотов соответствуют типам синтактико-семантических связей с их атрибутами, которые устанавливаются между словами синтаксическим анализатором текста.

Изначально фреймовое представление эквивалентно представлению в форме сети синтактико-семантических отношений [16], но в ходе работы СИ сеть пополняется новыми фреймами, не соответствующими никаким словам исходного текста, а также новыми связями между уже существующими и новыми фреймами, причем исходные фреймы вместе со своими связями могут удаляться из сети.

Например, фраза *Из карабина деда барон послал пулю в дикую утку* после синтаксического анализа преобразуется в следующую сеть связанных фреймов:

БАРОН, ДЕД, ПУЛЯ, ДИКИЙ– элементарные фреймы, не имеющие слотов.

КАРАБИН { #_Gen|Владелец=ДЕД }

ПОСЫЛАТЬ { #_Nom|Субъект = БАРОН, #_Acc|Объект = ПУЛЯ, #в_Acc|Цель|Место = УТКА, #из_Gen|Обстоятельство = КАРАБИН }

УТКА { Attribute=ДИКИЙ }

Здесь каждая строка начинается с имени фрейма, после которого в фигурных скобках перечисляются его слоты в форме СписокИменСлота = ИмяФреймаВСлоте. Осмысленные имена слотов вида Цель|Место устанавливаются на основе данных, указанных в словаре моделей управления предикатов естественного языка, и могут вообще отсутствовать. Альтернативу им представляют универсальные имена слотов, которые устанавливаются на основе общего алгоритма преобразования синтактико-семантических отношений [16] и присутствуют обязательно, например: Attribute (тип связи), #из_Gen (#предлог_ПадежСвязи типа "управление с предлогом"), #_Acc (#_ПадежСвязи типа "управление без предлога").

Применение определенного правила СИ приведет к добавлению в сеть нового фрейма с полностью "осмысленными" ролями слотов: СТРЕЛЬБА { Субъект|Стрелок = БАРОН, Объект|Цель = УТКА, Обстоятельство|Оружие = КАРАБИН, Обстоятельство|Снаряд = ПУЛЯ }, что представляет собой семантическую интерпретацию фразы. Дополнительно, правило позволяет удалить из сети проинтерпретированный фрейм ПОСЫЛАТЬ, осуществив таким образом полную семантическую трансформацию.

Семантические трансформации пропозиций и язык их описания

Пропозиция, по [17], – это языковый способ описания ситуации с характерной конфигурацией синтаксических связей, аналог глубинно-синтаксической структуры по [7].

Примеры различных пропозиций, выражающих одну и ту же ситуацию: *X решил проблему*. *X сумел решить проблему*. *X добился успеха в решении проблемы. X сумел добиться успеха в решении проблемы. X использовал свой шанс добиться успеха в решении проблемы.* Первая пропозиция является базовой и выражена т.н. изосемически изоморфной конструкцией. Заметим, что вариации *X решил проблему*, *проблема решена X-м*, *решившему проблему X-у* относятся к той же самой пропозиции, реализованной различными поверхностно-синтаксическими способами, и после синтаксического анализа и т.н. постсинтаксических трансформаций связей [16] будут представлены полностью идентичными сетями фреймов.

Правила семантической трансформации пропозиций позволяют приводить сколь угодно сложные пропозиции к базовым. Ключевая идея подхода заключается в том, что для семантической интерпретации любой пропозиции в требуемый фрейм достаточно описать трансформации всех пар входящих в нее элементарных пропозиций и применять их итерационно к результату предшествующих трансформаций до тех пор, пока сеть фреймов не перестанет трансформироваться. Полный набор правил трансформаций элементарных пропозиций обеспечивает корректную, с точки зрения здравого смысла, интерпретацию любой пропозиции, допустимой в языке. Так, представленные выше пропозиции могут быть трансформированы к базовой применением следующих трансформаций элементарных пропозиций: *X сумел "сделать" -> X "сделал"*, *X добился успеха в Y -> X преуспел в Y*, X *преуспел в решении* Y -> X *решил* Y, *X использовал шанс "сделать" -> X "сделал"*, где *"сделать"* условно обозначает любой глагол в инфинитной форме.

Типичное правило семантической трансформации имеет следующий вид:

```
verb { name=посылать|послать|отправлять|отправить } : Объект = пуля
-> стрельба { Субъект|Стрелок = verb:Субъект, Объект|Цель = verb:#в_Acc,
Обстоятельство|Оружие = verb:#из_Gen, Обстоятельство|Снаряд = пуля } &
del(verb)
```

Символ '->' отделяет левую часть правила, в которой задаются ограничения на пару связанных фреймов, от правой, в которой задаются действия по трансформации сети фреймов в том ее фрагменте, который удовлетворяет ограничениям из левой части. В левой части правила `Объект` задает имя слота, с которым фрейм `пуля` должен входить в связанный фрейм с атрибутом `name=посылать`, либо с другими перечисленными через оператор '|' именами. Псевдоним `verb`, произвольно выбранный для этого фрейма, используется чтобы однозначно сослаться на него в правой части правила, так как он, в отличие от фрейма `ПУЛЯ`, допускает множество альтернативных имен. В правой части `стрельба` задает имя нового создаваемого фрейма, а записи в фигурных скобках вида `Имя1Слота|Имя2Слота=ИмяФрейма:ИмяСлота` задают способ заполнения слотов нового

фрейма: все фреймы, входящие во фрейм `ИмяФрейма` из левой части правила в качестве слотов в роли `ИмяСлота`, войдут в новый фрейм как слоты с именами `Имя1Слота|Имя2Слота`. Оператор `del()` инициирует удаление заданного фрейма-аргумента.

Формально синтаксис языка семантических трансформаций можно описать так:

```
FrameLeft:ИмяСлота=FrameLeft -> FrameRight (& FrameRight)* (& ParentExpression)* (& del(ИмяФрейма))*, где:

FrameLeft::= ИмяФрейма / ПсевдонимФрейма { Атрибут=Значение(|Значение)* (, Атрибут=Значение(|Значение)*)* }

FrameRight::= FrameName { TransformRule (,TransformRule)* }

TransformRule::= (ИмяСлота(|ИмяСлота)* = FrameName / FrameName:ИмяСлота) / (ИмяАтрибута=$Значение)

FrameName::= ИмяФрейма / ПсевдонимФрейма

ParentExpression = parent(FrameName) { ИмяСлота(|ИмяСлота)* = FrameName }
```

Здесь кириллицей обозначены элементарные выражения - произвольные текстовые строки, а латиницей - сложные выражения, которые раскрываются в последующих строчках после символа '::='. Другие специальные символы относятся к описанию формата правил: '/' разделяет альтернативные выражения, допустимые в одной позиции, '*' обозначает возможность повторения выражения от нуля до произвольного количества раз, круглые скобки группируют выражения для применения к ним '/' и '*' как к единому целому. Левая часть правила, до символа '->', описывает условия, которым должна удовлетворять пара связанных фреймов, при соблюдении которых выполняются трансформации, описанные в правой части, после символа '->'.

Элементарные условия на атрибуты фрейма задаются в левой части правила в вида `Атрибут=Значение(|Значение)*`, где возможные значения атрибута с именем `Атрибут` разделяются оператором альтернативы '|'. В случае указания нескольких элементарных условий через запятую, комплексное условие в левой части считается выполненным в том случае, если выполнены все элементарные условия. Помимо точного сравнения атрибута с заданными значениями посредством оператора '=', в языке поддерживается сравнение с семантикой "значение не равно ни одному из заданных альтернативных" посредством оператора '!=', а также сравнение по подстроке с маской '*' - например, условие `semantic=`

Object* будет выполнено для любых значений атрибута semantic, начинающихся с префикса Object типа Object:Animated, Object:Inanimated:~Mental.

Каждое подвыражение TransformRule в правой части правила описывает действие по заполнению слота фрейма с указанными именами ИмяСлота ссылкой на фрейм, обозначенный как FrameName; или ссылкой на фрейм в его слоте, обозначенный как FrameName:ИмяСлота. Подвыражение вида ИмяАтрибута=$Значение трактуется как добавление атрибута с заданным ИмяАтрибута и Значение.

FrameName в составе выражения FrameRight в правой части может соответствовать одному из двух ИмяФрейма (или ПсевдонимФрейма) в выражениях FrameLeft в левой части правила - тогда указанные в подвыражениих TransformRule действия по заполнению слотов или добавлению атрибутов производятся с уже существующим фреймом. Если FrameName в правой части не совпадает ни с одним из ИмяФрейма или ПсевдонимФрейма в левой части, то в сети создается новый фрейм с именем FrameName, к которому и применяются действия, описанные в подвыражениях TransformRule.

ПсевдонимФрейма вместо ИмяФрейма в выражении FrameLeft используется тогда, когда правилом допускается несколько имен у фрейма - тогда таковые перечисляются как альтернативные значения атрибута name в фигурных скобках, как в приведенном выше примере правила: verb { name=посылать|послать|отправлять|отправить }. Введение псевдонима фрейма в случае множества альтернативных имен обеспечивает однозначность указания на фрейм в правой части правила: Субъект|Стрелок = verb:Субъект. ПсевдонимФрейма может иметь особый вид: frame или frameЛюбойТекст, что указывает на фрейм с любым именем, для которого при необходимости в фигурных скобках могут быть заданы дополнительные ограничения на прочие атрибуты. Ниже пример соответствующего правила, которое добавляет в сеть фреймов новый фрейм ДЕД с атрибутами semantic=_Noun и semantic=Object:Animated, а затем добавляет его в существующий frame в качестве слота с именем Владелец, в итоге преобразуя пропозиции вида *дедовский "предмет"* в "*предмет" деда*:

```
frame:Атрибут = дедовский -> дед { semantic=$_Noun, semantic=$Object:Animated } & frame { Владелец="дед" }
```

В правой части правила вместо ИмяСлота может использоваться символ * (не отражено в формальном описании синтаксиса во избежание путаницы со служебным символом *, обозначающим возможность произвольного повторения выражения в правиле),

что означает обращение ко всем слотам фрейма. Ниже пример соответствующего правила, которое добавляет во фрейм frame_*verb* глагола в инфинитиве все слоты фрейма *verb*_service с их именами, в итоге преобразуя пропозиции вида "*субъект" "так-то" начал "действовать"* в *"субъект" "так-то" действует,* и добавляя служебный глагол verb_service во фрейм полнозначного глагола frame_verb в качестве слота с именем Модус:

```
verb_service { name=закончить|окончить|завершить } :#_Inf = frame_verb { semantic=_Verb } -> frame_verb { *= verb_service:*, Модус=verb_service }
```

Подвыражение ParentExpression в правой части правила означает, что фрейм FrameName из части ИмяСлота(|ИмяСлота)* = FrameName войдет в слоты всех фреймов-родителей, содержащих фрейм FrameName из части parent(FrameName) в своих слотах, с теми же именами. Это иллюстрирует правило, которое преобразует пропозиции типа *написать (новую) серию статей* в *написать (новые) статьи*, *размышлять над (огромным) множеством проблем* в *размышлять над (огромными) проблемами*:

```
noun { name=ряд|серия|множество|масса|куча } :#_Gen = frame -> frame { *= noun:* } & parent(noun) { *=frame }
```

Сначала правило добавляет все слоты фрейма noun (например, СЕРИЯ) в слоты фрейма frame *(*например, СТАТЬЯ*),* а затем во фрейм, содержащий фрейм noun (СЕРИЯ) в качестве слота, добавляет фрейм frame (СТАТЬЯ) в качестве слота с теми же именами. Фреймов-родителей, содержащих один и тот же фрейм noun в своих слотах, может быть несколько, например в случае однородных членов *написать и издать серию статей* или деепричастного оборота *написать серию статей, издав (их)* - тогда правило включит фрейм СТАТЬЯ в качестве слота во все фреймы-родители НАПИСАТЬ, ИЗДАТЬ.

Приложения семантического интерпретатора

Интерпретация описаний событий и фактов

Стандартной задачей современных систем извлечения знания из текста, к которым относят системы knowledge management (KM) и business intelligence (BI), системы сбора фактографической информации и ведения конкурентной разведки, является задача выделения в тексте описаний заданных классов ситуаций (событий или фактов), выраженных разными языковыми способами, в единообразной структурированной форме. Такой форме соответствует представление ситуаций фреймами, с которым работает СИ.

Первое из приведенных в статье правил семантических трансформаций позволяет интерпретировать пропозиции вида *Стрелок послал пулю в Цель из Оружия* как фрейм-ситуацию `СТРЕЛЬБА { Стрелок, Цель, Оружие }`.

Рассмотрим пример извлечения факта - фрейма `ЗАКЛЮЧЕНИЕ_ДОГОВОРА { Сторона, Договор, Содержание }`.

Правило 1 интерпретирует пропозиции вида *Сторона поставила подпись под Объектом* как *Сторона подписала Объект*:

```
v { name = поставить|ставить } :Объект = подпись -> подписывать { Субъект = v:Субъект, Объект = v:#под_Gen }
```

Правило 2 извлекает целевой факт в структурированной форме как фрейм `ЗАКЛЮЧЕНИЕ_ДОГОВОРА`, интерпретируя пропозиции вида *Сторона1 подписала Договор со Стороной2 о Содержании*; *Сторона1 и Сторона2 подписали договор о Содержании*, которые могли быть сформированы в результате трансформации сети фреймов Правилом 1, либо изначально присутствовать в исходной сети:

```
verb { name = подписать|подписывать } :Объект = contract { name = договор|соглашение|контракт } -> Заключение_Договора { Сторона = verb:Субъект, Сторона = contract:#c_Gen, Договор = contract, Содержание = contract:#o_Prep }
```

Правило 3

```
решить: #_Inf = frame_v -> frame_v { Субъект = решить:Субъект, Квантор = решить:Квантор },
```

дает возможность трансформировать пропозиции вида *Субъект решил "действовать"* к виду *Субъект "действует"*. Дополнительное выражение `Квантор = решить:Квантор` позволяет "притянуть" к фрейму `ДЕЙСТВОВАТЬ` слоты с именем `Квантор` от фрейма `РЕШИТЬ` в качестве возможных показателей отрицания: *Субъект не решил "действовать" -> Субъект не "действует"*.

Правило 4

```
принять:Объект = решение -> решить { Субъект = принять:Субъект, #_Inf = решение:#_Inf }
```

позволяет трансформировать пропозиции вида *Субъект "таким-то образом" принял решение "действовать"* к виду пропозиции, далее интерпретируемому Правилом 3: *Субъект "таким-то образом" решил "действовать"*.

В результате работы всех четырех правил будет проинтерпретирована как фрейм `ЗАКЛЮЧЕНИЕ_ДОГОВОРА` сложная пропозиция вида *Сторона1 приняла решение поставить подпись под Договором со Стороной2 о Содержании*.

Более оптимальный способ написания Правила 4 не создает в сети новый фрейм РЕШИТЬ, а просто добавляет к существующему фрейму РЕШЕНИЕ слот фрейма ПРИНЯТЬ с именем Субъект:

```
принять:Объект = решение -> решение { Субъект=принять:Субъект },
```

преобразуя пропозиции вида *Субъект принял решение "действовать"* в *решение Субъекта "действовать"*. Для интерпретации такой пропозиции достаточно расширить Правило 3:

```
sol { name=решить/решение } :#_Inf = frame_v -> frame_v { Субъект = sol:Субъект, Квантор = sol:Квантор }
```

Заметим, что если в правой части Правила 3 не указывать конкретные имена слотов Субъект и Квантор, а разрешить любые, указав *=решить:*, то семантическая трансформация в ряде случаев повысит полноту интерпретации: *безумное решение Субъекта "действовать" -> Субъект "действует" безумно,* а в ряде случаев приведет к искажению исходного смысла, понизив точность: *Субъект давно решил "действовать" -> Субъект давно "действует"*, что не эквивалентно. Соотношение приоритетов полноты и точности, определяемое особенностями прикладной задачи извлечения событий и фактов, определяет способы написания правил. Теоретически, язык семантических трансформаций позволяет достичь одновременно сколь угодно высоких полноты и точности интерпретации содержания текста путем написания правил, строго учитывающих все конкретные слова предметной области, что, однако, является трудоемкой задачей, сопоставимой с описанием значений слов в более "сильных" подходах к семантике (см. Введение).

Интерпретация оценочных высказываний

Задача извлечения из текста выраженных в нем оценок объектов (товаров, организаций, персон) становится все более востребованной в связи с масштабным развитием социальных сетей в Интернет, сообщения которых переполнены прямыми и косвенными характеристиками различных объектов, их действий и свойств. Лингвистическая специфика данной задачи была описана автором в [18]. Точная интерпретация оценок, выраженных в тексте нетривиальными синтаксическими конструкциями, в том числе со сложным образом выраженными отрицаниями, эффективно реализуется описываемым здесь механизмом СИ путем выделения фреймов специального вида.

Фреймы TONALITY выделяются при прямом выражении позитивной или негативной оценки следующими типами пропозиций: *Объект отстойный, Объект считается отстойным/отстоем, Объект отстой, Объект - отстойное "нечто", Объект "что-то делает" плохо,* где в позиции *Объект* также допускаются конструкции вида *"свойство" Объекта* (*политика Х-а, интерфейс Х-а*), а перечень конкретных слов, допустимых в

позиции *"свойство"*, задается для каждой целевой предметной области. Фреймы *TONALITY* могут содержать слоты со следующими именами:

- `EstimatedObject` – объект оценки и/или его свойство: то, что оценивают;
- `Estimation` – слово, выражающее оценку.

Аналогично выделяются оценки, в которых целевой объект выступает в роли синтаксического субъекта предиката, характеризующего своего субъекта позитивно или негативно: *Объект сломался, поломка Объекта, надежная работа Объекта*.

Правила семантических трансформаций для выделения фреймов `TONALITY` не содержат конкретных оценочных слов в позициях, выделяемых как слоты, что позволяет значительно уменьшить количество правил, необходимых для описания целевой предметной области. Так, количество универсальных оценочных слов, приложимых практически к любым объектам, в русском языке составляет несколько тысяч, а кроме них существует множество специальных слов, оценочная семантика которых меняется в зависимости от классов объектов и их свойств: *быстрая загрузка телефона* - позитив, *быстрая разрядка аккумулятора* - негатив. Поэтому для выделения фреймов `TONALITY` используется несколько общих правил, которые распознают все синтаксические конструкции, потенциально способные выражать оценочные пропозиции указанных выше типов, затем создают фреймы `TONALITY` и добавляют к ним в качестве слотов все слова, стоящие в позициях `EstimatedObject` и `Estimation`, например, все согласованные с существительным определения-прилагательные (*быстрый, но ненадежный мобильный интернет*):

```
frame_noun { semantic=_Noun } :Attribute { semantic=_Adjective }= frame_adj -> tonality { EstimatedObject=frame_noun, Estimation=frame_noun:Attribute }
```

Окончательно фреймы `TONALITY` обрабатываются в программном коде: слова-значения слотов `EstimatedObject` проверяются на вхождение в словари наименований целевых объектов и их свойств, а значения слотов `Estimation` - на вхождение в словари "универсальный позитив/негатив" или "специальный позитив/негатив для целевых объектов/свойств в позиции `EstimatedObject`".

Фреймы `SITUATION_EMOTION` и `SITUATION_ACTION` выделяются при описании в тексте таких ситуаций, которые характеризуют целевой объект либо через его отношение к другим сущностям, либо через отношение других сущностей к нему:

- `SITUATION_EMOTION` - характеризуют отношение синтаксического субъекта предиката к его синтаксическому объекту посредством выражения эмоций: *обожаю Объект, только идиоты испытывают восторг от Объекта* и наоборот: *Объект не любит*

пользователей, Объект довел меня до истерики. В ситуациях этого класса оценка объекта "позитив/негатив" зависит от оценки субъекта (*я, потребитель, народ*... - позитив; *идиот, чайник*... - негатив) и наоборот.

- `SITUATION_ACTION` - описывают определенное действие, которое приносит пользу или вред синтаксическому объекту предиката от синтаксического субъекта. В таких ситуациях оценка объекта не зависит от оценки субъекта: *я помогаю Объекту* - позитив, *эти негодяи оказывают помощь Объекту* - также позитив (для *Объекта* хорошо, кто бы ему не помогал), тогда как оценка субъекта, напротив, зависит от оценки объекта: *Объект принес мне пользу* - позитив, *Объект способствует коррупции* - негатив.

Фреймы `SITUATION_EMOTION` и `SITUATION_ACTION` могут содержать слоты:

- `EstimatedObject` – объект и/или его свойство, оценку которого описывает ситуация;
- `EstimatingSubject` – тот, кто оценивает или действует на объект в роли `EstimatedObject`;
- `EstimatedObjectPositive` – сущность, к которой объект в слоте `EstimatingSubject` относится положительно (*EstimatingSubject борется за EstimatedObjectPositive*);
- `EstimatedObjectNegaitive` – сущность, к которой объект в слоте `EstimatingSubject` относится отрицательно (*EstimatingSubject борется с/против EstimatedObjectNegaitive*);
- `Estimation` – слово, называющее ситуацию (*восторг, бороться*)

Окончательно эти фреймы обрабатываются в программном коде: оценка целевого объекта в слоте `EstimatedObject` определяется на основе оценочной семантики слова в роли `EstimationNegative` или `EstimationPositive`, в соответствии с классом ситуации - именем фрейма `SITUATION_EMOTION` или `SITUATION_ACTION`. Соответственно, для каждого класса целевых объектов формируются соответствующие словари позитивных и негативных сущностей. Например, для оценки политических деятелей в глазах народа позитивными являются сущности *пенсионер (пенсионеры ненавидят Объект), заработная плата, рабочее место (действия Объекта привели к росту заработной платы и открытию новых рабочих мест)*, а отрицательными – *олигарх, чиновник* (*олигархи и чиновники ненавидят Объект*), *безработица, инфляция (действия Объекта привели к росту безработицы и инфляции*).

Дополнительно существует класс правил, которые добавляют во все оценочные фреймы слоты с именем *Negation* - показатели отрицания для последующего инвертирования

оценки в программном коде. Например, для пяти пропозиций вида *Объект не плохой / не ломается / не вызывает восторг*; *у Объекта нет/мало недостатков*; *Объект перестал работать/сбоить*; *Объект утратил преимущества / устранил недостатки*; *хорошая работа объекта - миф* показатели отрицания будут "притянуты" в целевые оценочные `TONALITY` и `SITUATION` следующими правилами:

```
frame :Quantifier = frame_neg { name=не|ни|якобы } -> frame { Negation = frame_neg }

frame_no { name=нет|мало } :#_Gen = frame_n -> frame_n { Negation = frame_no }

v { name=перестать|прекратить|отказаться } :#_Inf = frame -> frame { Negation = v, Negation=v:Negation }

v { name=утратить|потерять|устранить } :#_Acc = frame -> frame { Negation = v, Negation=v:Negation }

frame_n { semantic=_Noun } :Apposition = frame_neg { name = ложь|неправда|обман|миф|россказни } -> frame_n { Negation = frame_neg, Negation = frame_neg:Negation }
```

Первое правило просто маркирует ряд элементарных отрицаний, синтаксически подчиненных соответствующим словам текста напрямую: находит частицы *не, ни, якобы* как слоты фреймов с именем `Quantifier` и добавляет этим слотам альтернативные имена `Negation`.

В последующих правилах слово, выражающее отрицание, добавляется в качестве слота `Negation` к соответствующему фрейму, а подвыражения `Negation=v:Negation` и `Negation=frame_neg:Negation` позволяют "притянуть" к этому фрейму дополнительные слоты-отрицания в пропозициях типа: *у Объекта якобы нет недостатков*, *Объект не прекращает сбоить*, *Объект не утратил достоинств*, *хорошая работа Объекта - не миф*.

Для интерпретации осложненных пропозиций типа *Объект якобы начал утрачивать преимущества* третьим и четвертым правилами необходимо наличие дополнительного правила, которое "притягивает" слоты-отрицания от вспомогательного глагола к управляемому им глаголу-инфинитиву:

```
v { name=начать|продолжить } :#_Inf =frame -> frame { frame:Субъект = v:Субъект, Negation=v:Negation }
```

Интерпретатор языка семантических трансформаций

Описанный в общих чертах выше семантический интерпретатор (СИ) представляет собой интерпретатор языка семантических трансформаций и работает следующим образом.

На этапе инициализации правил запускается парсер формального языка описания семантических трансформаций, который преобразует правила из текстовой формы во внутренние логические структуры данных для быстрого применения к сетям фреймов, а также строит индексные структуры для быстрого поиска возможных правил-кандидатов на применение к фреймам предложения на основе слов - имен фреймов, содержащихся в предложении и правилах.

Алгоритм СИ обеспечивает независимость результата применения правил семантических трансформаций от порядка их применения, который является произвольным в том смысле, что определяется исключительно внутренними сортировками структур данных для быстрого доступа. СИ обрабатывает текст последовательно по предложениям, применяя к сети фреймов предложения множество активных правил. В активное множество входят только те правила, которые либо содержат точное сравнение атрибута `name` со словами из предложения, либо содержат сравнения `name` с операторами *!=* и *, либо вообще не содержат сравнений атрибута `name` (ограничения установлены на другие атрибуты).

На первой итерации к сети фреймов предложения применяется каждое правило из множества активных и, в случае удовлетворения какой-либо парой связанных фреймов условиям в левой части правила, производится трансформация фреймов из этой пары или связанных с ними фреймов, описанная в правой части правила - добавление новых слотов или их имен, атрибутов. Уже существующие во фрейме слоты, их имена и значения атрибутов повторно не добавляются. Указатели на все фреймы, трансформированные правилами, а также указатели на фреймы, связанные с трансформированным входящими и исходящими связями, запоминаются как активное множество фреймов. На следующей итерации активные правила применяются только к активным фреймам, трансформируя их и связанные с ними фреймы с формированием нового активного множества фреймов для новой итерации применения активных правил. В норме процесс трансформации сети сходится: после некоторой итерации не оказывается трансформированных фреймов, и семантическая интерпретация предложения завершается (хотя возможно написать "особые" правила для бесконечной трансформации сети).

Чтобы обеспечить инвариантность результата трансформаций к порядку применения правил, а также избежать неустойчивости процесса трансформаций, удаление фреймов со всеми входящими и исходящими связями, описанное в правилах оператором `del()`, производится после завершения всех трансформаций, добавляющих информацию в сеть предложения.

В финале из сети фреймов могут быть удалены фреймы, полностью покрытые другими фреймами. Под покрытым понимается фрейм, все слоты которого входят в число

слотов какого-либо другого, покрывающего фрейма, а сам покрытый фрейм не входит в качестве слота ни в какие другие фреймы. Это позволяет исключить из результатов неполные, промежуточные интерпретации, которые входят в состав более полных. Например для фразы *замечательный Объект сломался* будет выделен оценочный фрейм `TONALITY { EstimatedObject = Объект, Estimation = замечательный }`, соответствующий подфразе *замечательный Объект*, который будет покрыт более полным оценочным фреймом `TONALITY { EstimatedObject = Объект, Estimation = замечательный, Estimation = сломаться }`, на основании которого и будет сформирована окончательная оценка высказывания - негатив (по логике принятия решения, наличие хотя бы одного негативного слота `Estimation` во фрейме `TONALITY` формирует негативную оценку).

Заключение

Описанный в статье семантический интерпретатор успешно использован в электронных сервисах мониторинга информации в социальных сетях, предоставляемых системой "Крибрум" (http://www.kribrum.ru), для выявления позитивных/негативных оценок объектов мониторинга, сбора фактов по объектам мониторинга и авторам сообщений. На момент участия автора в проекте “Крибрум” в 2015 году для интерпретации оценочных высказываний и фактов, несущих оценку объектов мониторинга, использовалось более 1700 правил семантических трансформаций, а для выделения более 20 классов событий и фактов, излагаемых авторами сообщений от первого лица и характеризующих "профиль" автора - более 400 правил. В настоящий момент автор развивает приложения семантического интерпретатора в проектах компании “ЭР СИ О” (http://www.rco.ru), где описанный подход был изначально рожден.

Литература

1. Schubert, L.K., Hwang, C.H. (2000): Episodic Logic meets Little Red Riding Hood: A comprehensive, natural representation for language understanding. In Iwanska, L., Shapiro, S.C. (eds.) Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language, MIT/AAAI Press, Menlo Park, CA, and Cambridge, MA, pp. 111-174.
2. (Das et al 2014) Das D., Chen D., Martins A. F. T., Schneider N., Smith N. A. Frame-Semantic Parsing // Computational Linguistics. 2014. V. 40. 1. 4. P. 9-56.
3. (Banarescu et al 2013) Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., Schneider, N.: Abstract Meaning Representation for Sembanking. In: Proceedings of the 7th ACL Linguistic Annotation

Workshop and Interoperability with Discourse, Sofia, Bulgaria, August 8-9, 2013 (2013)(www.aclweb.org/anthology/W13-2322; проверено 12.03.2016)

4. Clark A., Fox C., Lappin S. The Handbook of Computational Linguistics and Natural Language Processing. Wiley-Blackwell, 2010. – 802 p.
5. Dowty D.R., Wall R.E., Peters S. Introduction to Montague Semantics. Kluwer Academic Publishers, 1981. - 316 p.
6. Апресян Ю.Д. Лексическая семантика. М.: Наука, 1974, 366 с.
7. Мельчук И.А. Опыт теории лингвистических моделей "Смысл ⇔ Текст". М.: Наука, 1974. - 314 с.
8. Apresjan, J.D., Boguslavsky, I.M., Iomdin, L.L., Tsinman, L.L. Lexical Functions in NLP: Possible Uses // Computational Linguistics for the New Millenium: Divergence or Synergy? Proceedings of the International Symposium held at the Ruprecht-Karls-Universität Heidelberg. Festschrift in Honour of Peter Hellwig on the occasion of his 60th Birthday. 21-22 July, 2000. Manfred Klenner, Henriëtte Visser (Eds.). Heidelberg: Peter Lang, 2002. - P. 55-72.
9. Тузов В.А. Компьютерная семантика русского языка. - СПб.: Изд-во СПбГУ, 2003. — 400 с.
10. Bos J. A Survey of Computational Semantics: Representation, Inference and Knowledge in Wide-Coverage Text Understanding // Language and Linguistics Compass. – 2011. 5/6. – P. 336–366.
11. Lappin S., Fox C. The Handbook of Contemporary Semantic Theory. John Wiley & Sons, 2015. – 704 p.
12. Фомичев В. А. Математические основы представления смысла текстов для разработки лингвистических информационных технологий. Часть I // Информационные технологии. 2002. № 10. С. 16–25.; Часть II // Информационные технологии. 2002. № 11. С. 34–45.
13. Фомичев В. А. Формализация проектирования лингвистических процессоров. М. : МАКС Пресс. 2005. - 368 с.
14. Fomichov V. A. Semantics-Oriented Natural Language Processing: Mathematical Models and Algorithms. New York, Dordrecht, Heidelberg, London: Springer, 2010. - 354 p.
15. Фомичев В.А. Математические основы представления содержания посланий компьютерных интеллектуальных агентов. М.: ГУ-ВШЭ, издательство «ТЕИС», 2007. - 176 с.
16. Ермаков А.Е., Плешко В.В. Семантическая интерпретация в системах компьютерного анализа текста // Информационные технологии. - 2009. – N 6. – С. 2-7.

17. Всеволодова М.В., Деменьтева О.Ю. Проблемы синтаксической парадигматики: коммуникативная парадигма предложений. – М.: КРОН-ПРЕСС, 1997. – 176 с.
18. Ермаков А.Е., Киселев С.Л. Лингвистическая модель для компьютерного анализа тональности публикаций СМИ // Компьютерная лингвистика и интеллектуальные технологии: труды Международной конференции Диалог'2005. – Москва, Наука, 2005. - С.131-135.