Язык описания правил в системе лексического анализа Ея-текстов DICTASCOPE TOKENIZER

The language for describing rules in DICTASCOPE TOKENIZER — a system for lexical analysis of natural language texts

Скатов Д. С. (ds@dictum.ru), **Ливерко С. В.** (sl@dictum.ru),
Вдовина Н. А. (vn@dictum.ru), **Окатьев В. В.** (oka@dictum.ru)

ООО «Диктум», Нижний Новгород, Россия

В статье обсуждается лексический анализ ЕЯ-текстов в контексте задач извлечения информации (именованных сущностей, гипертекстовых переходов, терминологии) на основе правил. Язык DSTL, предлагаемый для их решения, обладает разнообразными выразительными средствами при высокой компактности описаний.

1. Введение

Решения для извлечения структурированных данных из неразмеченных ЕЯ-текстов востребованы во множестве приложений. Можно строить прогнозы, автоматически извлекая из новостных потоков упоминания персон, организаций, адресов, торговых марок и отношений между ними с учетом временно́й динамики; автоматизировать построение баз данных, находя в текстах наименования, коды, характеристики товаров; автоматически добавлять к существующим веб-страницам семантическую разметку.

Для решения всех этих задач традиционно используются механизмы применения *правил*, составляемых экспертом (возможно, в полуавтоматическом режиме). По правилу механизм определяет в тексте фрагмент с данными и извлекает их в определенном формате. База правил должна допускать пополнение и поддержку с адекватными трудозатратами. Для практического использования к механизму предъявляются требования по скорости: недопустим комбинаторный рост времени анализа при линейном росте количества правил.

В данной работе представлен язык описания правил DSTL (DictaScope Tokenizer Language), используемый в системе лексического анализа ЕЯ-текстов DictaScope Tokenizer (DST, от англ. *to tokenize* — снабжать метками, помечать). Он разрабатывался для создания быстрого и относительно простого в применении механизма лексического анализа, включающего функции уже известных разработок. Средства, находящиеся в открытом доступе [3, 12], не позволяли достичь требуемых свойств механизма, поэтому язык создавался независимо от них. Был учтен опыт специалистов, занимающихся решением схожих задач, и результаты собственных исследований [13].

2. Постановка задачи

Любая задача, затрагиваемая в данной работе, может быть сформулирована следующим образом: выявить в *неразмеченном* ЕЯ-тексте *лексические конструкции* — цепочки *слов* входного текста (*возможно, разрывные*), каждая из которых снабжается *набором данных* определенной структуры. Структура включает:

- имя *класса*, которому принадлежит конструкция;
- *нормальную форму конструкции*, которая состоит из *нормализованного текстового представления* (удобного для прочтения человеком) и набора именованных полей с присвоенными значениями.

Эта задача далее называется *лексическим анализом* [15] *естественного языка*, или, сокращенно — *LANL* (Lexical analysis of natural language, [1]).

В данной постановке может быть сформулирована известная задача *извлечения именованных сущностей* (NER — *Named Entity Recognition*, [5, 6, 9, 11]), к которым относят имена персон, организации, географические адреса, даты, результаты измерений

и пр. Для таких сущностей характерна вычислимая нормальная форма, а упоминания об одной сущности записываются в тексте в разных формах и являются *непрерывными* символьными конструкциями. В качестве классов выступают типы сущностей (*«Дата»*, *«Персона»*, *«Организация»*), а нормальная форма — это набор полей. Напр., запись «***31 июля 1986 г.***» (равно как и записи «***1986/31/7***», «***July 31st, 1986***») естественно отнести к классу *«Дата»* и снабдить нормальной формой в виде трех числовых полей *{Day = 31, Month = 7, Year = 1986}* и текстового представления «***31.07.1986***».

В материалах о приложении текстовых шаблонов к обработке ЕЯ эти шаблоны обозначаются поразному: встречаются лексические [4], семантические [6], лексико-синтаксические [12] шаблоны. Задачу NER часто относят к семантическому анализу [14]. Пересекаются понятия лексического анализа, токенизации, графематики. Для ясности изложения авторы далее уточняют интерпретацию этих терминов.

В LANL конструкции формируются из слов. Задачу разбиения текста на слова называют (символьной) токенизацией [19]. В теории формальных языков токенизация и лексический анализ — синонимы [15]. В обработке ЕЯ это, вообще говоря, не так. В публикациях по теме: (1) токенизация — это разбиение цепочек символов на слова [19], (2) лексический анализ — формирование конструкций из цепочек слов [1, 4]; выход (1) является входом (2). Решение этой задачи представляет самостоятельный интерес: она сложна для арабского, японского языков, любого текста с опечатками типа пропуска пробелов. В статье эта задача детально не рассматривается. Модель слова, принятая в языке DSTL, допускает адаптацию к любой схеме символьной токенизации.

При извлечении конструкций полезно выявить *отношения* между ними — такова основная цель *семантического анализа текста*. Эти отношения могут быть описаны лексическими конструкциями, которые, в отличие от именованных сущностей, чаще всего разрывны. Напр., *«Персона»* могла в прошлом являться сотрудником *«Организации»* — это образует *факт «Место работы»*: «***Василий Петров**, мечтая о научной карьере, долгое время успешно **трудился в НИИ ЧАВО***». Существенные фрагменты выделены жирным, они образуют утверждение вида *«X трудился в Y»*, а деепричастный оборот и уточнения для указанного факта несущественны.

Подходы, применяемые для выявления отношений и по сути решающие задачу LANL, используют элементы синтаксического анализа ЕЯ:

- применяется доступ к грамматическим значениям,
- явно присутствует дерево непосредственных составляющих (напр.: *«[[19 января 2010 г.]*$_{\text{Дата}}$ *вступил в должность [Заместитель [председателя [правительства РФ]* $_{\text{Организация}}$ *]* $_{\text{Должность}}$ *[Хлопонин А. Г.]*$_{\text{Персона}}$ *]*$_{\text{Должность}}$*]* $_{\text{Назначение}}$*»*).

Для выявления отношений требуется покрыть конструкциями лишь отдельные фрагменты текста; современный синтаксический анализ решает более широкую задачу — построение полного дерева предложения [13], по этой причине сложность его на порядки выше. Чтобы избежать оговорок, задачу извлечения отношений *на основе шаблонов* авторы работы также относят к LANL, без какого-либо упоминания о синтаксисе.

Полное представление о роли LANL в решении задач анализа ЕЯ дано на Рис. 1.

Следующие задачи, отличные от NER, также сводятся к LANL:

Отыскание в неразмеченном документе фрагментов, содержащих ссылки на другие документы и собственные разделы [10]: *«согласно ст.15 Конституции РФ»*;

Выявление в ЕЯ-текстах фраз-определений авторских терминов, их синонимов и связанных атрибутов [12]: «*Синтаксический анализ — это …*»;

Нормализация слабоструктурированных источников данных: автоматизированное формирование и коррекция номенклатурных списков (имущества, оборудования и т. д.) [4];

Т. н. прошивка законодательства — извлечение инструкций (связанных с обновлением текстов во времени) для их последующего применения [10]: «*Часть первую статьи 41 дополнить словами “или его заместителем”*».

Графематический анализ: к нему относят выявление в тексте простых лексических конструкций (ФИО с инициалами, электронные адреса, имена файлов), а также предложений, абзацев, заголовков, примечаний [7].

Анализ искусственных языков, как правило, осуществляется механизмами на основе формальных грамматик [14]. Их возможности в задачах анализа ЕЯ ограничены. В то же время, в рамках LANL они остаются эффективным описательным средством при условии соответствующих доработок, которые и были выполнены рядом исследователей [1–4, 6, 8–12, 17].

3. Обзор

Все известные авторам средства, решающие задач LANL с помощью шаблонов, основаны на регулярных выражениях. В 90-х был предложен инструмент NLlex [1], расширяющий Unix-утилиту lex доступом к морфологическому словарю и лингвистическим деталям текста. Тогда же был предложен язык CPSL [2]: в нем шаблон представляет собой решающее правило, в левой части которого записано регулярное выражение относительно свойств слов, в правой — действие, исполняемое при его срабатывании. Шаблоны CPSL можно наследовать. Этот

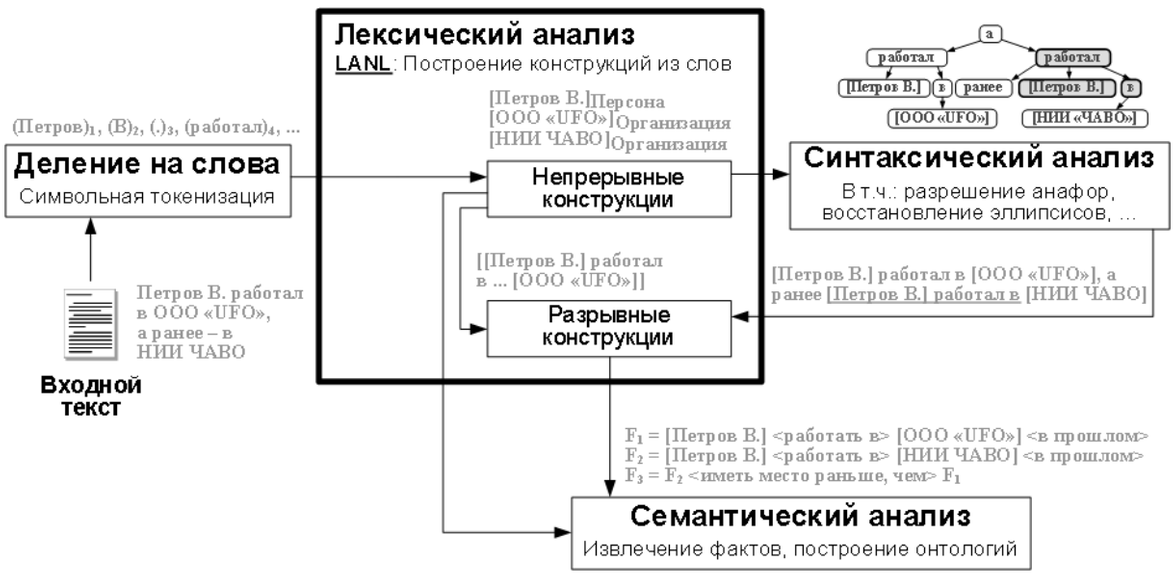

Рис. 1. Роль задачи LANL в общей схеме анализа ЕЯ

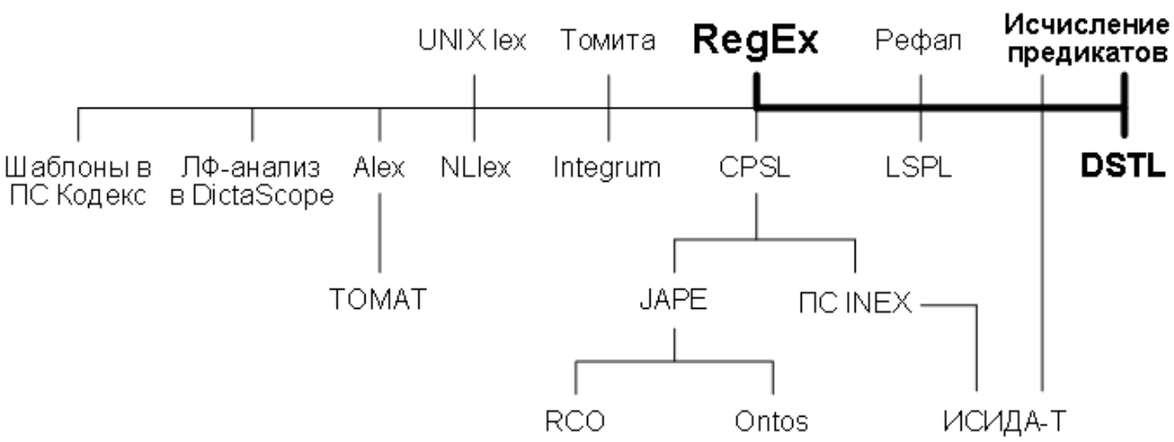

Рис. 2. «Генеалогическое древо» языковых средств для описания лексических шаблонов

язык перенял недостаток обычных регулярных выражений (regex-ов): их восприятие затруднено из-за большого числа парных скобок, а в CPSL это усугубляется наличием в шаблоне разных типов скобок. В языке JAPE [3] к CPSL была добавлена атрибутно-объектная модель текста для применения в рамках языка JAVA. Ни CPSL, ни JAPE не предусматривают проверок синтаксического согласования.

Для анализа русского языка на базе CPSL был создан язык в ПС INEX [8], на базе JAPE — языки в системах RCO Pattern Extractor [6] (ориентирован на извлечение именованных сущностей) и Ontos-Miner [9] (анализ новостных текстов). В системе «ИСИДА-Т» [17] возможности CPSL улучшены: расширены типы данных, можно создавать поля с произвольными значениями, формулировать проверки в виде логических предикатов (в частности — проверять согласование).

Ряд отечественных разработок вне ветви развития CPSL предназначен для решения частных задач: напр., ИПС «Кодекс» [10], Alex [4] (продолжила развитие в системе TOMAT). Функция т. н. лексико-фрагментационного анализа (разбиение текста на предложения с выявлением регулярных элементов типа дат и URL) встроена в синтаксический анализатор ЕЯ DictaScope [13]. Все это суть расширения regex-ов, но, в отличие от NLlex, они не учитывают морфологию.

За рамками статьи оставлено решение задач LANL методами машинного обучения. Этот подход широко применяется зарубежными исследователями в задаче NER [5], а в России представлен разработками IXLab [18]. Также можно отметить подход системы Integrum [11] (выявление именованных сущностей и их отношений), использующий алгоритм Томиты.

В работе [12] предложен язык LSPL с компактными лексическими шаблонами с наследованием и проверкой согласования. Он был применен создателями для выявления терминов в научной литературе, т. о. эти шаблоны изначально ориентированы на ограниченный тип конструкций [16]. Важные функции, свойственные CPSL, напр., работа с вычислимыми полями, в LSPL отсутствуют. В нем используются элементы, заимствованные из функционального программирования, поэтому синтаксис LSPL весьма необычен.

В силу требований гибкости и скорости ни одна из названных разработок не была положена в основу языка DSTL. Он построен на основе регулярных выражений и формализма исчисления предикатов. DSTL идейно и функционально близок к ИСИДА-Т (наиболее «продвинутому» потомку CPSL) и языку LSPL (при создании которого, согласно [12], принимались во внимание практически все языки из обзора). Можно утверждать, что DSTL покрывает возможности этих разработок.

4. Язык DSTL

4.1. Лексические правила

Словом в DSTL по умолчанию является цифробуквенное сочетание (*«январь»*, *«25»*), одиночный разделитель («.», «–») или подряд идущие пробельные символы. *Атом* — это цепочка из одного и более таких слов, первое и последнее из которых — непробельные: *«январь»*, *«несмотря на то что»* — атомы, *« 23 »* — не атом. Эта схема позволяет эксперту не беспокоиться о таких деталях, как наличие пробелов между словами и их количество (DSTL позволяет учесть их при необходимости).

Правило DSTL образовано несколькими секциями. Напр., правило с именем `Year` описывает конструкции вида *«1986 г.»*:

```
Year { /* 1986 г. { year = 1986; } */
   T := Y "г." ?;
   C := Length (Y) = 4 & IsNumeric (Y);
   A := { year := Y; };
};
```

Шаблон (секция с именем **T** — *template*) — это регулярное выражение, записанное относительно его *элементов* — имен атомов и унаследованных конструкций. В шаблоне правила `Year` утверждается: конструкция состоит из атома `Y` и двухсловного атома `"г."`, который может и отсутствовать (согласно оператору `?`).

Свойства `Y` формулируются в *критерии* (секция **C** — *criterion*). По сути — это предикат, составленный из доступных в DSTL функций с аргументами-элементами шаблона **T**. Доступны логические, арифметические, строковые функции. В секции **C** примера сказано: (1) атом `Y` состоит из четырех символов (`Length (Y) = 4`), (2) `Y` — число (`IsNumeric (Y)`).

Действие (секция **A** — *action*) выполняется, если критерий сработал (т. е. оказался истинным). Оно представляет собой последовательность операций. В действии примера строковая запись года присваивается полю `year`.

4.2. Схема работы с морфологией

В DST *грамматическое значение* (*ГЗ*) представляет собой набор *грамматических характеристик* единицы языка. К ним относятся грамматические категории (часть речи, род, число, падеж, время, …) и начальная форма слова. DST не снимает омонимию — в нем слово *«для»* будет иметь два ГЗ, соответствующие глаголу *«длить»* и предлогу *«для»*. Поэтому для единицы языка приходится оперировать *множеством* ГЗ. Лексическим конструкциям также присваивается ГЗ, поэтому на согласование можно проверить не только отдельные атомы, но и (1) атомы и конструкции, (2) пары конструкций.

Проверить наличие ГЗ с заданными характеристиками можно функцией `HasGrammarForm`. При этом строится т. н. *сужение множества ГЗ*: в рамках текущей конструкции остаются только ГЗ элемента с заданными характеристиками. Напр., множество ГЗ слова `W = "Александра"` образовано тремя элементами; они имеют одинаковые значения типа, подтипа и числа `{Type: Noun, Subtype: Name, Number: Sg}`, но различаются по роду и падежу: `W.GrV = [{Case: Nom, Gender: Fem} (ж. р. им. п.), {Case: Gen, Gender: Masc} (м. р. род. п.), {Case: Acc, Gender: Masc} (м. р. вин. п.)]`. Если описываются конструкции женских имен, нужно задать проверку `F(W) = HasGrammarForm (Name, {Type: Noun, Subtype: Name, Gender: Fem})`. Тогда, `F(W) = true` и `W.GrV = [{Case: Nom, Gender: Fem}]` (ГЗ с мужским родом оказываются отфильтрованными).

Два (и более) элемента можно проверить на согласование — этой цели служит функция `AreConcordant`. Она находит среди ГЗ первого и второго элемента пару значений, у которых совпадают заданные характеристики. Напр., если `V = "студентка"`, `W = "Александра"`, то `AreConcordant (V, W, {Number, Gender, Case})` проверит согласование `V` и `W` по числу, роду и падежу, результатом будет `true`, а при `V = "автомобиль"` результат — `false`. Если ранее функция `HasGrammarForm` построила сужение для `V` или/и `W`, то при проверке будут учитываться только ГЗ из этих сужений.

Схема, принятая в DSTL для обработки ГЗ, показана на Рис. 3. Фактически, она дает способ снятия омонимии по контексту. На последующих этапах обработки конструкций (напр., синтаксическим анализатором) для них будет известно конкретное ГЗ, выведенное лексическими правилами языка DSTL.

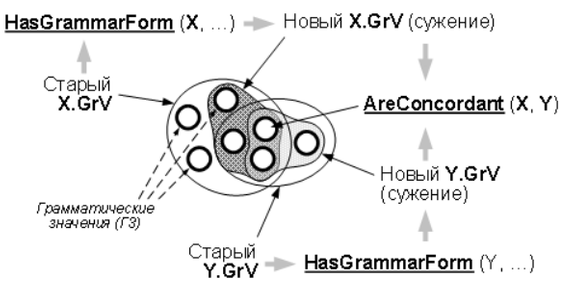

Рис. 3. Схема работы с грамматическими значениями в DSTL

4.3. Наследование конструкций

По структуре база правил в DSTL представляет собой КС-грамматику [15], в которой терминалы — это атомы, нетерминалы — имена правил. Операция использования одной конструкции в другой называется *наследованием*. Она подразумевает не только подстановку шаблона, но также и возможность обращаться к полям вложенной конструкции и наследовать их.

Далее правило `Year` (см. выше) снабжается дополнительными правилами для выделения дат: результирующая база описывает конструкции вида *«31 июля 1986 г.»*.

```
Months := { "января": 1, … ,"декабря": 12 };

Day {
   T := D; /* 31 {day: 31} */
   C := IsNumeric (D) & DiapStr (D, 1, 31);
   A := { day := StrToInt (D); };
};

Month {
   T := M; /* июль, июля {month: 7} */
   C := M in Months; /* Months["июля"] = 7 */
   A := { month := Months[M]; };
};

/* 31 июля {day: 31, month: 7} */
Day_Month { T := [Day] [Month]; };
/* 31 июля 1986 г. {day: 31, month: 7, year: 1986} */
Date { T := [Day_Month] [Year]; };
```

В строке *«31 июля 1986 г.»* можно определить три значащие конструкции: в силу правил `Day_Month`, `Year` и `Date`. Т. к. `Date` наследует `Day_Month` и `Year`, результатом анализа является *дерево* с этими конструкциями в вершинах. Набор таких деревьев, узлами которых являются все конструкции текста, называется *лексическим покрытием* (уточнение понятия *аннотаций* [17]).

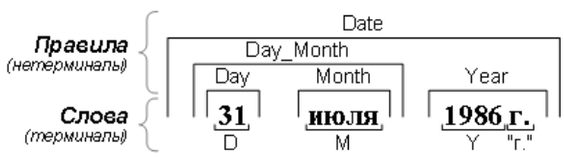

Рис. 4. Лексическое покрытие текста конструкциями класса «Дата»

При наследовании возможны конфликты конструкций, напр.: (1) *«Александра Иванова»* может быть персоной женского рода в им. п., а может — мужского в род. п., (2) во фрагменте *«Пушкин А. С. Поэмы»* могут быть определены персоны *«Пушкин А. С.»* и *«А. С. Поэмы»*, (3) для *«заместителя управляющего делами президента РФ Павла Бородина»* возможны три варианта покрытия (см. Рис. 5).

Рис. 5. Три различных дерева, соответствующие одной конструкции

Для разрешения конфликтов в DST существует система *вычислимых* весов, но эта тема выходит за рамки статьи.

4.4. Согласование и нормальная форма

Следующий пример позволяет извлечь упоминания персон мужского рода (ед. и мн. ч.) вида *«Петровым Сергеем Николаевичем»*, *«Иваны Петровы»* и т. д. Определяется правило `N` для имени:

```
N {
   T := W; /* Иван, Петру, Сергеем */
   C := HasGrammarForm (W, {Subtype: Name, Gender: Masc});
   A := { GrV := W.GrV; W := GetInitialForm (W); };
};
```

Затем аналогично `N` строятся правила `Sn` (фамилия) и `Pt` (отчество) и описываются персоны:

```
N_Sn { T := [N] [Sn]; /* Иванам Петровым */
   C := AreConcordant (N, Sn, {Gender, Number, Case}); };
Sn_N { T := [Sn] [N]; /* Петрову Ивану */
   C := AreConcordant (N, Sn, {Gender, Number, Case}); };
N_Pt_Sn { T := [N] [Pt] [Sn]; /* Ивана Михайловича Петрова */
   C := AreConcordant (N, Pt, Sn, {Gender, Number, Case}); };
Sn_N_Pt { T := [Sn] [N] [Pt]; /* Петровым Иваном Михайловичем */
   C := AreConcordant (Sn, N, Pt, {Gender, Number, Case}); };
```

Т. к. проверяется согласование, то в тексте *«Иван Петрова не видел»* персоны *«Иван Петрова»* (`N_Sn`) найдено не будет, но найдутся *«Иван»* (`N`) и *«Петрова»* (`Sn`).

4.5. Сравнение языков

CPSL был выбран для сравнения как наиболее «плодовитый» представитель шаблонных языковых средств (см. Рис. 2), близкий по функциональности к DSTL. Средства без поддержки морфологии [1, 4] по возможностям далеки от DSTL. LSPL, несмотря на идейную близость к DSTL, представляет другой крайний случай — все его существенные возможности сосредоточены исключительно в морфологии. Следующие примеры взяты в [8].

В этом сравнении, по мнению авторов, прослеживается бóльшая выразительность DSTL.

5. Алгоритм анализа

Далее дается краткое описание подхода к анализу, который применяется в DST.

Пусть $A=\{a_1,\ldots,a_N\}$ — набор атрибутов, таких, что для произвольного слова V $a_k(V)\in\{0,1\}$. Для каждого слова W, которое должно попасть в конструкцию, определяемую правилом R, из шаблона и критерия выявляется множество атрибутов $\alpha(W)\subseteq A$, значения которых для этого слова W должны быть истинными (значения остальных атрибутов для W не важны).

В шаблон вместо имени любого его слова W подставляется множество $\alpha(W)$. Далее полученный шаблон T трактуется как обычное регулярное выражение, в котором вместо символов рассматриваются множества $\alpha(W)$, вместо отношения равенства двух символов (W из шаблона и V из проверяемого текста S) — отношение вхождения подмножества во множество: $W\subseteq V$.

Пример. Пусть $A=\{1,2,3\}$, $T=\{1,2\}\{3\}$, $S=\{1,2\}\{1,2,3\}\{2,3\}\{3\}$. Шаблон T входит в текст S в смещении 1 (т. к. $\{1,2\}\subseteq\{1,2\}$, $\{3\}\subseteq\{1,2,3\}$) и 2 (т. к. $\{1,2\}\subseteq\{1,2,3\}$, $\{3\}\subseteq\{2,3\}$), но не в 3 (т. к. $\{1,2\}\not\subseteq\{2,3\}$).

Пример 1: Извлечение IP-адресов вида *«192.168.1.72»*.

CPSL

```
Rule: IPAddress (
   {Token.kind == number} {Token.string == "."}
   {Token.kind == number} {Token.string == "."}
   {Token.kind == number} {Token.string == "."}
   {Token.kind == number})
: ipAddress --> :ipAddress.Address = {
     kind = "ipAddress")
```

DSTL

```
IPAddress {
   T := N ("." N) {3};
   C := IsNumeric (N);
   A := { kind := "ipAddress" };
};
```

Пример 2: Извлечение имен печатных СМИ: *«Газета “Веселый огородник”»*, *«Журнал “Smog”»*.

CPSL

```
Macro: NOT_QUOTE ({!Token.string == "\""})
Rule: NewspaperName
   ({Token.string =| "газета"} | {Token.string =| "журнал"})
   {Token.string == "\""}
   (({!Token.string == "\"", Morpho.Capitalized == True})
    NOT_QUOTE? NOT_QUOTE? NOT_QUOTE?)
    : newspaperName {Token.string == "\""}
--> :newspaperName.ProperName = {
   kind = "Newspaper", rule = "NewspaperName"}
```

DSTL

```
QUOTE := "\"";

Name : hidden {
   T := First (Other) {0,3};
   C := IsCapitalized (First) &
        First != QUOTE & Other != QUOTE;
};

Newspaper {
   T := Pr QUOTE [Name] QUOTE;
   C := Pr %in {"газета", "журнал"};
   A := { kind := "Newspaper"; newspaperName := Name; };
};
```

6. Заключение

Язык DSTL представляет собой новую ветвь развития современных средств описания ЕЯ-шаблонов. Разработчики учитывали достоинства и недостатки существующих разработок, чтобы создать емкий и выразительный язык, который сохранил бы полезную функциональность этих средств, но освободил составителя решающих правил от известных проблем. К ним относятся высокая сложность вложенных конструкций в регулярных выражениях и необходимость учета лишних деталей, таких как особенности объектной модели текста [3] или наличие пробелов между словами [1].

Признано, что регулярные выражения обладают большой выразительностью, но их экземпляры из реальной жизни нельзя назвать понятными. Назначение языка DSTL — устранить сложность regex-ов в задачах анализа ЕЯ, сделав описания достаточно короткими и естественными, отражающими лингвистическую суть конструкций. Описание структуры шаблона отделено от фиксации свойств его элементов, что, по мнению авторов, предотвращает разрастание шаблонов и размытие их смысла и, тем самым, упрощает расширение и поддержку правил.

Язык DSTL используется авторами для решения задач LANL, которые были обозначены в начале статьи. На данный момент можно утверждать, что в плоскости извлечения именованных сущностей язык достиг устойчивой фазы: имеются достаточно качественно функционирующие описания классов *«Персона»*, *«Организация»*, *«Время и дата»* и пр. Напр., извлечение объектов типа *«Персона»* выполняется на машине Athlon 3,1 GHz в объеме памяти 50 Мб со следующими усредненными по корпусу текстов из WikiPedia показателями: скорость 700 Кб/с, точность ~ 0,8; полнота ~ 0,9.

Ведутся исследования по применению DSTL для извлечения фактов на основе разрывных конструкций для извлечения фактов. В ближайших планах — исследования по автоматизированному составлению решающих правил на основе методов машинного обучения.

Литература

1. *Almeida J. J.* Nllex — a tool to generate lexical analysers for natural language. Technical Report UMDI95.04 // Universidade do Minho, Departamento de Informatica: 1995.
2. *Appelt D., Onyshkevych B.* The Common Pattern Specification Language // Annual Meeting of the ACL. Proceedings of a workshop on held at Baltimore, Maryland, October 13–15. Association for Computational Linguistics, Morristown, NJ, USA: 1998. P. 23–30.
3. *Cunningham H., Maynard D., Tablan V.* JAPE: A Java Annotations Pattern Engine. Technical Report CS-00-10 // University of Sheffield. UK: 2000.
4. *Жигалов В. А., Жигалов Д. В., Жуков А. А., Кононенко И. С., Соколова Е. Г., Толдова С. Ю.* Система Alex как средство для многоцелевой автоматизированной обработки текстов // Труды международного семинара Диалог-2002. М.: Наука, 2002. Т. 2. С. 192–208.
5. *Bender O., Och F. J., Ney H.* Maximum Entropy Models for Named Entity Recognition // CoNLL-2003, 7th Conference on Computational Natural Language Learning. Edmonton, Canada: May 2003. P. 148–152.
6. *Ермаков А. Е., Плешко В. В., Митюнин В. А.* RCO Pattern Extractor: компонент выделения особых объектов в тексте // Информатизация и информационная безопасность правоохранительных органов: XI Международная научная конференция. Сборник трудов. М.: 2003.
7. *Ножов И. М.* Морфологическая и синтаксическая обработка текста (модели и программы) // Интернет-публикация авторской диссертации на сайте www.aot.ru. М.: 2003.
8. *Программная* система извлечения информации из текстов (ПС INEX). Программная документация // 04832915.10028-01 33 01. М.: 2004.
9. *Дудчук Ф. И., Шафирин А. Ю.* Извлечение информации из франкоязычных текстов: морфология, синтаксис, объекты // Девятая Национальная конференция по искусственному интеллекту с международным участием КИИ-2004: Труды конференции. В 3-х т. Т. 2. М.: Физматлит, 2004. С. 489–497.
10. *Губин М. В., Меркулов А. И.* Автоматическое выделение гипертекстовых переходов в текстах документов // Труды международной конференции «Диалог-2004». М.: Наука, 2004. С. 155–158.
11. *Гершензон Л. М., Ножов И. М., Панкратов Д. В.* Система извлечения и поиска структурированной информации из больших текстовых массивов СМИ. Архитектурные и лингвистические особенности // Труды международной конференции «Диалог-2005». М.: Наука, 2005.
12. *Большакова Е. И., Баева Н. В., Бордаченкова Е. А., Васильева Н. Э., Морозов С. С.* Лексико-синтаксические шаблоны в задачах автоматической обработки текста // Труды международной конференции «Диалог 2007». М.: Изд-во РГГУ, 2007. С. 70–76.
13. *Окатьев В. В., Гергель В. П., Алексеев В. Е., Таланов В. А., Баркалов К. А., Скатов Д. С., Ерехинская Т. Н., Котов А. Е., Титова А. С.* Отчет о выполнении НИОКР по теме: «Разработка пилотной версии системы синтаксического анализа русского языка» (инвентарный номер ВНТИЦ 02200803750) // М.: ВНИТЦ, 2008.
14. *Азарова И. В., Гребеньков А. С., Ландо Т. М.* Использование маркеров актантных позиций при анализе деловых текстов для расширения логической схемы предметной области // Труды международной конференции «Диалог 2008». М.: РГГУ, 2008. С. 11–17.
15. *Ахо А. В., Лам М. С., Сети Р., Ульман Д. Д.* Компиляторы: принципы, технологии и инструментарий, 2 изд. М.: «Вильямс», 2008.
16. *Рабчевский Е., Булатова Г., Шарафутдинов И.* Формализм записи лексико-синтаксических шаблонов в задаче автоматизации процесса построения онтологий // Труды десятой всероссийской научной конференции «RCDL'2008». Дубна: ОИЯИ, 2008. С. 415.
17. *Кормалев Д. А., Куршев Е. П., Сулейманова Е. А., Трофимов И. В.* Извлечение информации из текста в системе ИСИДА-Т // Труды 11-й Всероссийской научной конференции RCDL'2009. Петрозаводск: 2009. С. 247–253.
18. *Алексеев С. С., Морозов В. В., Симаков К. В.* Методы машинного обучения в задачах извлечения информации из текстов по эталону // Электронные библиотеки: перспективные методы и технологии, электронные коллекции: Труды XI-ой всероссийской научной конференции (RCDL'2009). Петрозаводск: КарНЦ РАН, 2009. С. 237–246
19. *Bird S., Klein E., Loper E.* Natural Language Processing with Python. USA: O'Reilly, 2009.