ИНФОРМАТИКА

2007 **январь-март** **№ 1**

УДК 004.89:004.03

А.В. Чеусов

АЛГОРИТМ СИНТАКСИЧЕСКОГО АНАЛИЗА ТЕКСТА ДЛЯ ОБРАБОТКИ БОЛЬШИХ ОБЪЕМОВ ДАННЫХ

Описывается эффективный алгоритм реализации синтаксического анализа естественного языка. Представленный алгоритм используется в промышленной системе инженерии знаний и применяется для анализа английского, японского, французского и немецкого языков. Эффективность алгоритма объясняется линейной зависимостью времени его работы от длины предложения.

Введение

В эпоху интенсивной информатизации практически всех сфер человеческой деятельности задача интеллектуализации информационного поиска и инженерии знаний является одной из важнейших задач, требующих эффективного решения. Универсальным средством описания действительности и коммуникации с вычислительной системой является естественный язык (ЕЯ), поэтому интеллектуализация информационных систем требует привлечения развитых лингвистических процессоров (ЛП), в чьи задачи входит как обработка текстов на этапе наполнения базы данных или базы знаний, так и обработка пользовательского запроса к ним. К ЛП в составе информационно-поисковых систем предъявляются жесткие требования как по качеству лингвистической функциональности, так и по скорости работы всех его модулей. Последнее особенно важно, поскольку, во-первых, значительны объемы лингвистических ресурсов, привлекаемых в составе баз знаний, во-вторых, достаточно велики объемы документов даже в специализированных информационных системах (нередко речь идет о десятках и сотнях гигабайтов), не говоря уже о сети Интернет, где объемы значительно выше и постоянно имеет место приток новых документов.

Традиционно лингвистические процессоры реализуют следующие основные этапы обработки ЕЯ: лексический анализ, лексико-грамматический, синтаксический и семантический. Настоящая работа посвящена синтаксическому анализу − одному из наиболее сложных и важных с указанных выше точек зрения этапов обработки ЕЯ. Безусловно, существует достаточно много различных методов и алгоритмов решения задачи синтаксического анализа текста, но все еще остаются актуальными в рамках этой задачи проблемы точности и скорости обработки произвольных текстов. Даже наиболее известные лингвистические модули, такие как описанные в работе [1], не обладают необходимой на сегодняшний день производительностью, не говоря уже о том, что многие публикуемые показатели были получены разработчиками на небольших объемах тестировочного материала, зачастую даже предварительно отредактированного. Именно с этой точки зрения и рассматривается задача синтаксического анализа текста (САТ) в данной работе. Входом для модуля САТ является, как правило, аннотированное лексико-грамматическими классами (иногда называемыми тегами) предложение, параграф либо весь текст документа, при этом аннотирование предполагает или полное разрешение омонимии, когда каждому слову текста ставится в соответствие единственный лексико-грамматический тег, или частичное, когда некоторые слова остаются многозначными. Например, при полном разрешении омонимии английское предложение «The table is red.» может быть аннотировано следующим образом: «The_ATI table_NN is_BEZ red_JJ ._.». Знак подчеркивания здесь отделяет слово от его лексико-грамматического класса, при этом знаки препинания также рассматриваются в качестве слов и им приписываются соответствующие классы. Для аннотирования слов в этом примере используется классификатор известного аннотированного корпуса текстов LOB [2].

Под задачей САТ понимают, как правило, либо построение дерева синтаксического вывода [3], либо выделение только определенных заранее заданных типов отношений между словами текста [4].

1. Аппарат расширенных регулярных выражений WRE

Описываемый в настоящей работе подход к синтаксическому анализу текста основан на расширенных регулярных выражениях WRE [5]. Ниже приводится их формальное определение. Пусть T^{POS} − конечное непустое множество лексико-грамматических классов слов ЕЯ, а T^{SEM} − конечное непустое множество семантических классов слов. Обозначим символом W множество последовательностей алфавитно-цифровых символов, символов знаков препинания и т. п. Таким образом, множество W включает все возможные слова ЕЯ, знаки препинания, числительные, даты и др. Определим функции $D_0^{POS} : T^{POS} \to 2^W$ и $D_0^{SEM} : T \to 2^W$, которые ставят в соответствие каждому лексико-грамматическому и семантическому классу соответственно множество слов (здесь 2^W означает множество всех подмножеств W). Например, D_0^{POS} ставит в соответствие лексико-грамматическому классу **NN** (существительное нарицательное в единственном числе) множество английских слов «table», «apple», «book» и т. п., а классу **VB** (начальная форма глагола) − множество слов «comprise», «perform», «translate» и др. Заметим, что множество слов для отдельных классов может быть бесконечным (как, например, бесконечно количество числительных, записанных арабскими цифрами), поэтому данные функции обычно задаются не только с помощью словарей ЕЯ, т. е. с помощью списков слов фиксированного размера, но и с помощью специальных алгоритмов распознавания лексико-грамматических свойств слова, таких, например, как описанный в работе [6]. Далее определим $D^{POS}(x) = D_0^{POS}(x) \times T^{POS}$ для всех $x \in T^{POS}$ и $D^{SEM}(x) = D_0^{SEM}(x) \times T^{POS}$ для всех $x \in T^{SEM}$ для отображения множеств лексико-грамматических и семантических классов соответственно во множество слов, аннотированных произвольными лексико-грамматическими классами согласно функциям D_0^{POS} и D_0^{SEM}, здесь знак × означает декартово произведение множеств. Например, $D^{POS}(NN)$ ={(«table», AT), («table», DT), («table», NN), …, («apple», AT), («apple», DT), …}. Также обозначим множество всех слов, аннотированных заданным классом слов, как $K(x) = W \times \{x\}$ для всех $x \in T^{POS}$ и множество всех слов, аннотированных всеми лексико-грамматическими классами, как $A = W \times T^{POS}$. Кроме того, зададим функцию $E_0 : R \to 2^W$, где R − множество обычных «символьных» расширенных регулярных выражений (extended regular expression, ERE) [7], широко применяемых в UNIX-подобных операционных системах и средах. Множество E_0 ставит в соответствие регулярному выражению множество соответствующих ему слов, например E_0(*«ing$»*)={*«thing», «consisting», «being», ...*}. Далее зададим $E(r) = E_0(r) \times T^{POS}$ для всех $r \in R$.

Важно отметить, что функции D^{POS}, D^{SEM}, K, E, по сути, накладывают различные «шаблоны» на аннотированные слова. Комбинируя эти шаблоны, можно легко задавать произвольные множества аннотированных слов, пользуясь для этого операциями над множествами. Например, $D^{POS}(VBG) \cap E(«ing\$») \cap K(NN)$ задает множество слов, аннотированных классом NN, заканчивающихся на «ing» и омонимичных с первым причастием согласно лексико-грамматическому словарю, а множество $K(VB) \cap \overline{D}^{POS}(NN) \cap D^{SEM}(LinkVerb)$ задает множество слов, аннотированных классом VB, не омонимичных с существительным и помеченных в семантическом словаре классом LinkVerb, здесь $\overline{D^{POS}}(NN) = A \setminus D^{POS}(NN)$. Необходимость в задании подобного рода шаблонов в задачах обработки ЕЯ возникает довольно часто, и задача синтаксического анализа здесь не является исключением.

Наконец, дадим рекурсивное определение регулярному множеству WRE, которое задает последовательности аннотированных слов:

− пустое множество является регулярным множеством WRE;

− множество последовательностей, которые состоят из одного слова множества A, является регулярным множеством WRE;

− множество последовательностей, состоящих из одного слова множеств $D^{POS}(t^{pos})$, $D^{SEM}(t^{sem})$, $K(t^{pos})$ или $E(r)$, является регулярным множеством WRE;

− множество последовательностей, состоящих из одного слова множества, заданного операциями пересечения и отрицания (вычитание из A) над множествами $D^{POS}(t^{pos})$, $D^{SEM}(t^{sem})$, $K(t^{pos})$ или $E(r)$, является регулярным множеством WRE;

− если R_1 и R_2 – регулярные множества WRE, то $R_1 \cup R_2$ и $R_1R_2 = \{r1^* \; r2 |\; r1 \in R_1, r2 \in R_2$ − регулярные множества WRE, здесь знак $*$ означает операцию конкатенации последовательностей аннотированных слов.

Расширенным регулярным выражением WRE называется грамматика, язык которой представляет собой регулярное множество WRE. Для ее записи используется специальная нотация, во многом схожая с той, которая применяется для записи ERE в среде программирования UNIX. Например, **^ DO|DOZ|DOD !DO!HV!VB!BE + $** задает множество аннотированных предложений, в которых первое слово аннотировано классами DO, DOZ или DOD (эти классы соответствуют словам «do», «does» и «did»), за которым вплоть до последнего слова предложения нет ни одного слова, аннотированного начальной формой глагола, т. е. классами HV, DO, VB или BE. Или, например, **VBZ|VBD|VB/!@LinkVerb [«not»] (JJ/'able$' | NN/'able$') IN|TO** задает множество аннотированных предложений, в которых присутствует следующая последовательность слов: любая форма смыслового глагола, не помеченная в словаре семантическим классом LinkVerb, слово «not» (может отсутствовать), прилагательное или существительное, заканчивающееся на «able», любой предлог или частица «to». Подробное описание синтаксиса можно найти в работе [5], здесь лишь отметим, что программный модуль, реализующий расширенные регулярные множества WRE, является важнейшим компонентом разработанного ЛП.

2. Распознавание именных словосочетаний в тексте

Особую роль в лингвистическом анализе текста на ЕЯ играет выделение именных словосочетаний (ИС). Эта задача имеет ряд важных приложений, например, в системах информационного поиска, кластеризации и классификации документов, машинного перевода и инженерии знаний, автоматического реферирования и т. д. Дело в том, что именные словосочетания являются своего рода «заготовками» для построения ключевых слов, дескрипторов, концептов и пр., т. е. базовых элементов индексирования текстовых документов и запросов пользователя.

Грамматика выделения ИС разрабатывается на основе существующих знаний о ЕЯ, дополненных результатами анализа больших по объему корпусов текстов, а каждое ее правило представляется в виде расширенного регулярного выражения WRE [5]. В нашем случае это множество правил R может быть определено следующим образом: $R = (R_1, R_2, \ldots, R_M)$, где R_i – правило грамматики, которое задает множество именных словосочетаний определенного типа и, в общем случае, оперирует лексическими единицами и их лексико-грамматическими, синтаксическими и семантическими классами; M – общее количество правил. Язык рассматриваемой грамматики ($L(R)$) будет представлять собой множество аннотированных именных словосочетаний ЕЯ всех типов − объединение языков каждого правила, т. е. $L(R) = \bigcup_{i=1}^{M} L(R_i)$.

Например, два из наиболее простых правил выделения именного словосочетания английского языка будут выглядеть следующим образом:

AT|ATI|DT ? OD|JJ * NN
ATI|DTS ? OD|JJ * NNS.

Как и в предыдущем примере, здесь использован лексико-грамматический классификатор LOB. В соответствии с ним NN и NNS – это лексико-грамматические классы существительного

единственного числа и множественного соответственно, AT – класс для неопределенных артиклей «a» и «an», ATI – класс определенного артикля «the», DT, DTS – классы детерминативов единственного числа «this», «that» и других и множественного числа «these», «those» и других, OD, JJ – классы порядковых числительных и прилагательных соответственно. Приведенные правила задают в качестве именного словосочетания последовательность слов, начинающуюся с артикля либо детерминатива (они могут и отсутствовать), за которыми следует последовательность (возможно, пустая) из прилагательных или порядковых числительных, за которыми, в свою очередь, следует существительное единственного (первое правило) либо множественного (второе правило) числа. Таким образом, входом для модуля распознавания ИС является, с одной стороны, аннотированное лексико-грамматическими классами предложение, т. е. предложение, в котором каждому слову поставлен в соответствие единственный здесь лексико-грамматический класс, а с другой − грамматика распознавания именных словосочетаний, заданная правилами из множества R.

В работе [8] показано, что для любой грамматики WRE существует эквивалентный ей конечный автомат (КА) [9]. Значит, используя детерминированный конечный автомат (ДКА), можно за линейное время выяснить, удовлетворяет ли заданная цепочка аннотированных слов текста грамматике, заданной R. Используя этот факт, сформулируем алгоритм выделения именных словосочетаний в предложении. Начиная с первого слова, ищем «самую левую самую длинную» [10] цепочку слов, удовлетворяющую грамматике. В случае успеха помечаем найденную цепочку как именное словосочетание и продолжаем поиск, начиная со слова, следующего за последним словом найденной цепочки. Если цепочка не найдена, продолжаем поиск, начиная со второго, третьего и так далее вплоть до последнего слова в предложении.

Теоретически трудоемкость данного алгоритма – $O(N^2)$, где N – количество слов в предложении. Необходимо отметить, что на практике скорость распознавания ИС с помощью описанного выше алгоритма не только не зависит от количества правил (благодаря использованию ДКА), но и имеет в среднем линейную зависимость от длины предложения, т. е. среднюю трудоемкость алгоритма можно оценить как $O(N)$, что, безусловно, характеризует алгоритм как достаточно эффективный. Однако, как показали теоретические исследования, оценку трудоемкости $O(N^2)$ алгоритма (в худшем случае) можно улучшить, модифицировав его следующим образом. Для набора правил $R = (R_1, R_2, \ldots, R_M)$ построим регулярное выражение WRE $R^* = .^*(R_1 \mid R_2 \mid \ldots \mid R_M)$, язык которого $L(R^*)$ равен множеству последовательностей слов, заканчивающихся именным словосочетанием. Здесь точка (в соответствии с[5]) обозначает любой символ входного алфавита (в нашем случае – любое аннотированное слово из A), а звездочка − звезду Клини [9, 10], т. е. произвольное количество повторений предыдущего символа или регулярного выражения. Таким образом, пара символов «.*» (точка, звездочка) обозначает последовательность аннотированных слов любой длины (возможно, пустую). Также построим $R^{rev} = (R_1^{rev}, R_2^{rev}, \ldots, R_M^{rev})$, состоящую из множества «перевернутых» правил из R (имеется в виду, что язык $L(R_i^{rev})$ равен множеству последовательностей аннотированных слов из языка $L(R_i)$, записанных в обратном порядке). Теперь поиск именных словосочетаний произведем следующим образом. Начиная с первого слова в предложении, согласно R^* ищем цепочку слов, которая заканчивается именным словосочетанием. Если таких цепочек нет, то алгоритм завершает работу, в противном случае последнее слово найденной цепочки является последним словом первого в предложении именного словосочетания. Далее, начиная с последнего слова найденной цепочки, т. е. с последнего слова найденного ИС, с помощью R^{rev} ищем начало ИС, анализируя слова по направлению к началу предложения. Так находим первое слово именного словосочетания. Найденная цепочка указывает на начало первого именного словосочетания в предложении. Для поиска остальных ИС повторяем описанные выше шаги, начиная поиск со слова, следующего за последним словом найденного ИС.

Данный алгоритм требует не более двух просмотров каждого слова в предложении. Следовательно, трудоемкость алгоритма в худшем случае составляет $O(N)$. Необходимо отметить, что с ростом количества правил и их сложности размер ДКА, эквивалентный регулярному выражению WRE, растет экспоненциально. Это, вообще говоря, свойственно допускающим КА [9] (в описанном алгоритме быстрый рост размера ДКА особенно актуален для R^*) и не может не сказаться на требованиях к количеству оперативной памяти ЭВМ, необходимой для работы программы, которая реализует алгоритм. Поэтому выбор между предложенными алгоритмами выделения ИС зависит от конкретных требований к системе, особенностей реализации WRE (в частности, от алгоритма укладки разреженных матриц в ОЗУ) и аппаратных особенностей ЭВМ. Также отметим, что представленные алгоритмы позволяют не только выделить именное словосочетание, но и определить его тип, т. е. его грамматические категории, такие как число, род, падеж и др. Для этого необходимо поставить в соответствие каждому правилу из R тип именного словосочетания и вместо ДКА для R (или R^{rev}) использовать автомат Мура [9], состояниям которого поставлен в соответствие необходимый выходной вес, т. е. тип именного словосочетания.

3. Построение дерева синтаксического вывода

Выделив в предложении именные словосочетания, можно приступить к следующему, завершающему этапу синтаксического анализа, а именно к выделению отношений между различными именными группами и словами, не включенными ни в одну из них, например к выделению связи «подлежащее-сказуемое-дополнение» или «глагол-обстоятельство» и т. п. Эта задача в данном случае реализуется через построение дерева синтаксического вывода и так же, как и выделение именных словосочетаний, решается с помощью аппарата расширенных регулярных выражений WRE.

Итак, входом для данного этапа синтаксического анализа является последовательность аннотированных именных словосочетаний и слов, не попавших ни в одну из выделенных ИС, таких как глаголы, союзы, частицы и др. Например, в предложении «The_table_NN phrase is_BEZ red_JJ ._.» классом NN помечено именное словосочетание, а не отдельное слово. Дерево вывода строится согласно стратегии «снизу-вверх» [11], т. е. начиная с узлов более низкого уровня, которые затем объединяются в узлы более высокого уровня и т. д.

Соответствующая грамматика имеет вид $G = (T, N, s, P)$, где T – множество терминальных символов, т. е. лексико-грамматических классов и различных типов именных словосочетаний, N – множество нетерминальных символов, $s \in N$ – стартовый символ грамматики, $P = (P_1, P_2, \ldots, P_k)$ – набор правил вывода. Каждое правило P_i имеет вид $(C_l C C_r \to C_l n C_r)$, где $n \in N$; C_l, C_r и C – регулярные выражения WRE, классификатор которых равен $T \cup N$ (C_l и C_r задают левый и правый контексты для C соответственно). В целом, грамматика G подобна контекстно зависимой грамматике, но от последней ее отличают два существенных обстоятельства: жесткий порядок производимых объединений, описанный ниже, и тот факт, что вместо комбинации символов из N и T в правилах вывода применяются регулярные выражения WRE. Последнее отличие придает грамматике G значительно большую гибкость и выразительность. Объединения же производятся следующим образом. Начиная с конца предложения, находим «самую правую самую длинную» цепочку символов (аннотированных слов, именных словосочетаний или нетерминальных символов), подходящих под левую часть правила из P. Если необходимая цепочка не найдена, алгоритм завершает свою работу, в противном случае определяем границы левого и правого контекстов C_l и C_r, заменяем часть строки, соответствующую C, на нетерминальный символ n, после чего повторяем процедуру поиска.

Алгоритм поиска подходящей цепочки такой же, как и при распознавании именных словосочетаний, и, как было показано, имеет трудоемкость $O(N)$, поэтому, если минимальная

длина цепочки, подходящей под левую часть правил из P, не меньше двух (в этом случае после каждой замены входных символов на нетерминальный символ длина исходной цепочки уменьшается, по меньшей мере, на единицу), трудоемкость полученного алгоритма равна $O(N^2)$. Ограничивая длину цепочки, которая может быть найдена по правилу из P, до некоторой константы L (на практике это приводит к запрету на использование в регулярных выражениях WRE операции повторения *), несложно получить алгоритм с трудоемкостью в худшем случае $O(L{\cdot}N)$, т. е. фактически получить линейную зависимость времени обработки предложения от его длины. Для этого исходный алгоритм необходимо модифицировать следующим образом. После создания очередного узла дерева синтаксического вывода начинаем поиск следующей цепочки не с конца предложения, а со слова, отстоящего от вновь созданного узла на L позиций вправо, т. е. по направлению к концу предложения.

В случае успешного завершения работы алгоритма получим дерево вывода с корневым узлом S, по структуре которого можно восстановить типы отношений между терминальными символами, т. е. между отдельными словами и именными группами. Примеры деревьев вывода показаны на рис. 1 и 2. Листьями этих деревьев являются аннотированные слова, а узлами − нетерминальные символы грамматики, при этом узлы «предков» в дереве изображены выше узлов «потомков». Кроме того, о том, каким образом был выведен нетерминальный символ (узел в дереве), говорят узлы его непосредственных «потомков», например, нетерминальный символ VerbPhrase в предложении на рис. 1 был выведен по правилу грамматики VerbPhrase := VB CS Clause.

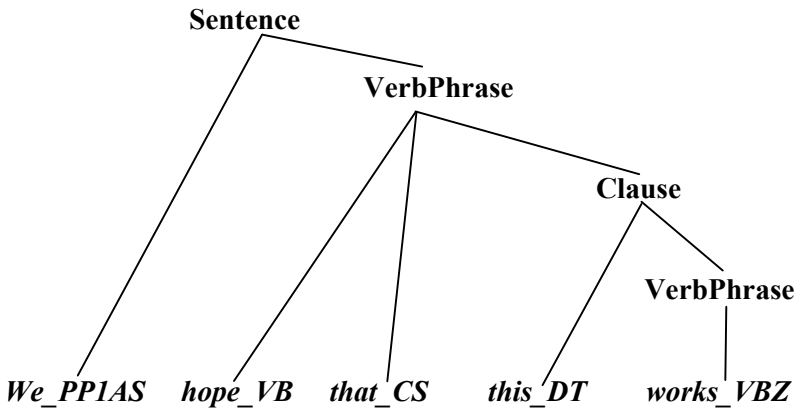

Рис. 1. Дерево вывода предложения «We hope that this works»

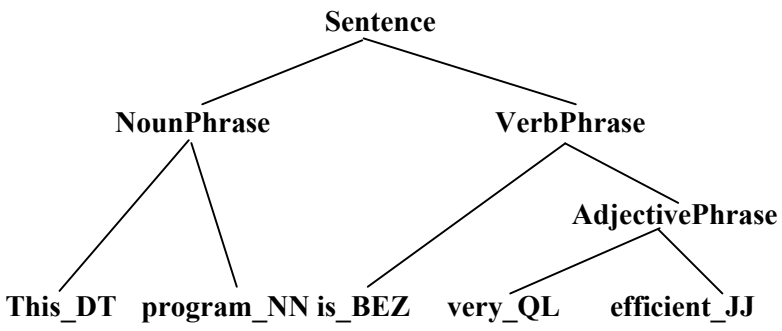

Рис. 2. Дерево вывода предложения «This program is very efficient»

Отметим, что грамматика синтаксического анализа для английского языка в нашем случае включает порядка 1500 правил вывода, а скорость анализа входного текста, например, на ЭВМ, оснащенной центральным процессором Pentium-4 с тактовой частотой 2800 МГц, составляет порядка 1,5–2 Мб/с.

Заключение

Представленные эффективные алгоритмы выделения именных словосочетаний из текстов и построения дерева синтаксического анализа были успешно реализованы в виде программных модулей и легли в основу лингвистического процессора промышленной системы инженерии знаний и автоматизации решения изобретательских задач [12], в рамках которой была продемонстрирована их высокая производительность, достаточная для обработки больших объемов документов на английском языке. Позже, в процессе эксплуатации и расширения функциональности лингвистического процессора, алгоритмы показали себя эффективными также для обработки французского, немецкого и японского языков. Это, в конечном счете, позволило создать высокопроизводительный многоязычный лингвистический процессор промышленной обработки произвольных текстовых документов.

Список литературы

1. Connexor natural language [Electronic resource]. − 2006. − Mode of access: http://www.conexor.fi.

2. The Tagged LOB Corpus: Users' manual / S. Johansson [et. al.]. − The Norwegian Computing Centre for the Humanities. − Norway: Bergen University, 1986.

3. Grishman, R. Computational linguistics. An introduction / R. Grishman. − United Kingdom: Cambridge University Press, 1986.

4. Link grammar parser [Electronic resource]. − 2006. − Mode of access: http://www.link.cs.cmu.edu/ link/submit-sentence-4.html.

5. Cheusov, A.V. The word-based regular expressions in computational linguistics / A.V. Cheusov // Proc. of the 7th International conference, Pattern Recognition and Information. – Minsk, 2003. − Vol. 1. − P. 208–213.

6. Mikheev, A. Automatic rule induction for unknown-word guessing / A. Mikheev // Computational Linguistics. – 1997. − Vol. 23, № 3. − P. 405–423.

7. IEEE Std 1003.1−2001. Standard for Information Technology – Portable Operating System Interface (POSIX) System Interfaces, Issue 6, 2001.

8. Postanogov, D. Effective implementation of word-based regular expressions notation in natural language processing / D. Postanogov // Proc. of the seventh International conference, Pattern recognition and information processing. – Minsk, 2003.

9. Брауэр, В. Введение в теорию конечных автоматов / В. Брауэр. − М.: Радио и связь, 1987.

10. Friedl, J. Mastering Regular Expressions / J. Friedl. − O'Reilly & Associates, Inc., 1997.

11. Jurafsky, D. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition / D. Juravsky, J. Martin. − New Jersey: Prentice Hall PTR, 2000.

12. Invention Machine [Electronic resource]. − 2006. − Mode of access: http://www.invention-machine.com.

Поступила 11.09.06

ИП «Инвеншион Машин»,
Минск, Революционная, 11
e-mail: vle@gmx.net,
cheusov@imb.invention-machine.com

A.V. Cheusov

ALGORITHM OF SYNTACTICAL TEXT ANALYSIS FOR LARGE DATA AMOUNT PROCESSING

An efficient algorithm of syntactical analysis of natural language is presented. The algorithm is used in well-known commercial knowledge engineering system. It is used for processing English, Japanese, French and German texts. The algorithm has a linear complexity from the sentence length.