ISBN 5-94052-066-0 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ. ПЕРЕСЛАВЛЬ-ЗАЛЕССКИЙ, 2004

УДК 519.767.6

Д. А. Кормалев, Е. П. Куршев, Е. А. Сулейманова,
И. В. Трофимов

Архитектура инструментальных средств систем извлечения информации из текстов

Аннотация. Статья посвящена описанию подходов и архитектурных решений, используемых при создании инструментальных средств для систем извлечения информации из текстов на естественном языке. Описаны общие подходы к извлечению информации, проанализированы потребности систем извлечения информации и их типичное устройство. Предлагается вариант архитектуры инструментальных средств, опробованный на практике.

Ключевые слова и фразы: извлечение информации, анализ текста, инструментальные средства.

Введение

Извлечение информации из текстов и представление ее в виде формальной системы знаний (например, в виде набора фреймов или семантической сети) — важная задача в области автоматической обработки текстов на естественном языке.

Извлечение информации (Information Extraction) [1] — это подход, который позволяет сузить круг задач, требующих специфического предметно-ориентированного решения при анализе текста. В рамках этого подхода задача обработки текста ограничена распознаванием множества классов ключевых понятий конкретной предметной области и игнорированием всякой другой информации.

Работа выполнена при поддержке программы фундаментальных исследований ОИТВС РАН «Фундаментальные основы информационных технологий и систем».

Рассмотрим систему фреймов [2], построенную на основе анализа текста и являющуюся формализованным представлением содержащихся в нем знаний. Каждому ее элементу соответствует определенный фрагмент текста. Возможно не прямое отображение, а в результате, например, логического вывода[1]. Слотам, отражающим одинаковые по смыслу данные (например, тип события, его участники, дата), обычно соответствуют сходные по структуре и/или контексту фрагменты исходного текста. Следовательно, можно определить правила, которые осуществляют сопоставление образцу (возможно, с учетом контекста) на уровне структуры части исходного текста и, в случае успешного сопоставления, строят новые или уточняют существующие фреймы. При описании структуры текста могут указываться условия (ограничения), накладываемые на каждое из слов[2], порядок их следования и связи различного вида между ними. Условия, упомянутые выше, могут накладываться как на стандартные лингвистические (графематические, морфологические, семантические, полученные на основе тезауруса) характеристики слов, так и на информацию об этом слове (фрагменте текста), полученную в результате выполнения других правил. Если какая-то часть текста удовлетворяет всем указанным в правиле условиям, то происходит ее формализация согласно соответствующей части правила. Следовательно, для технологии извлечения информации ключевым элементом является построение набора правил, позволяющих локализовать релевантную информацию в тексте.

Технология извлечения информации из текстов может служить основой для разработки средств анализа текста, оперирующих на более высоком уровне, например, интеллектуального анализа текста (*text mining*) [3] или наполнения реляционных или объектных баз данных. Помимо этих, несомненно важных, приложений технология извлечения информации может быть использована в целом спектре задач [4, 5]

Настоящая статья посвящена описанию подходов и архитектурных решений при создании систем извлечения информации. Сначала

[1]Изложение методов и средств для такого подхода лежит за пределами данной статьи, однако все средства, описанные здесь, применимы и для непрямого отображения.

[2]Можно обобщить этот подход для произвольного фрагмента текста, как показано в разделе 2.

мы обратимся к анализу потребностей систем извлечения информации (раздел 1), затем перейдем к общему описанию подхода и используемых методов (раздел 2). Далее рассмотрим типичную архитектуру систем извлечения информации (раздел 3). На основании анализа требований, с учетом выбранного подхода и типичной архитектуры систем извлечения информации перейдем собственно к описанию состава и архитектуры инструментальных средств (раздел 4). В заключение приведем несколько соображений по дальнейшим направлениям работ (раздел 5).

1. Анализ требований к системам извлечения информации

Автоматическая обработка текста на естественном языке позволяет облегчить поиск и извлечение информации с целью дальнейшей аналитической обработки. Чаще всего требуется анализ больших массивов коротких текстов (например, новостей) с целью выделения значимой информации. В качестве такой информации может выступать описание какого-либо события, его действующие лица, локализация в пространстве и времени.

Системы извлечения информации осуществляют обработку текста на разных уровнях:

- первичная фильтрация документа;
- лингвистическая обработка:
  - графематика;
  - морфология;
  - синтаксис;
- выделение простейших семантических структур;
- собственно извлечение информации;
- объединение построенных структур;
- разрешение анафоры и кореферентности.

В идеале система извлечения информации должна быть независимой от языка и предоставлять возможность настройки на произвольный естественный язык. Однако в данной статье речь пойдет в первую очередь про обработку текстов на русском языке.

Рассмотрим требования, выдвигаемые к инструментальным средствам. Они должны обеспечивать:

- минимально необходимый набор стандартных средств лингвистической обработки (из списка, приведенного выше);

- возможность замещения модулей и настройки порядка их применения;
- стандартные средства отладочного ввода-вывода;
- поддержку некоторого языка описания правил извлечения информации.

Ясно, что нельзя закладывать в качестве основных требований, например, средства разрешения кореферентности или объединения построенных структур. Подобные задачи имеют множество решений, каждое из которых лучше подходит в одной или другой предметной области. В частности, эксперименты с подходом к разрешению анафоры, описанным в [6], показали, что набор атрибутов (весовых коэффициентов), указывающих значимость каждого из возможных антецедентов, должен настраиваться индивидуально для каждой предметной области; а для некоторых предметных областей такой подход применим только с очень значительными ограничениями.

2. Описание подхода

Разработка целостной системы инструментальных средств требует единого подхода. Множество видов и этапов обработки текста может создать впечатление, что невозможно выразить лингвистическую информацию и информацию предметной области единообразно, но на самом деле это не так.

Для успешного извлечения информации из текста система должна располагать некоторой дополнительной информацией, которая не присутствует в тексте в явном виде. Речь идет о наборе атрибутов, приписанных фрагментам текста: морфологических, синтаксических, лексических, семантических и т. п. Для получения этой дополнительной информации о тексте производятся различные виды анализа текста. Анализ носит многоуровневый характер, поэтому можно считать, что лингвистический процессор состоит из набора анализаторов. Каждый анализатор исследует одну из лингвистических характеристик текста. Как правило, последующие уровни анализа текста используют результаты, полученные на предыдущих этапах[3]. Применение правил извлечения информации не является последним этапом, стоящим отдельно; мы рассматриваем его как часть прикладного семантического анализа.

[3]Деление на этапы довольно условно, поскольку часто требуется взаимодействие анализаторов.

2.1. Модель. Существует два основных подхода к представлению информации о тексте: *ссылочный* и *аддитивный*. Аддитивный подход подразумевает модификацию исходного текста с добавлением в него специальных служебных символов (вариант такого подхода — использование языка разметки XML). Преимущество очевидно — текст после обработки может быть сохранен, и в дальнейшем всю полученная информацию можно восстановить без повторной обработки. Недостатками этого метода являются необходимость модификации текста и пониженная, по сравнению с ссылочным подходом, производительность. Ссылочный подход предлагает хранение информации о тексте отдельно от самого текста и привязку к тексту с использованием ссылок. Производительность здесь, как правило, выше, но возникает необходимость поддержки специальных структур данных. В ссылочном подходе можно выделить два направления:

- создание фиксированной объектной модели текста, что приемлемо для конечных приложений обработки текста (однако при попытке обобщения проявляется нестабильность такой модели);
- создание унифицированного подхода, например, использование модели аннотаций, о которой рассказывается ниже.

В нашем подходе к выражению информации о фрагментах текста мы во многом следуем эталонной архитектуре *TIPSTER* [7]. Процесс получения дополнительной информации при анализе текста будем называть *аннотированием*. Любая лингвистическая (и другая) информация о тексте представляется в виде *аннотации*. Аннотация сопоставляется фрагменту текста, принадлежит классу аннотаций и обладает атрибутами. Класс аннотаций — это строка, позволяющая разбить все аннотации на смысловые группы. Для сопоставления аннотации фрагменту текста используется начало и длина (или начало и конец) фрагмента текста, к которому приписана аннотация. Атрибуты аннотаций представляют собой пары ⟨имя, значение⟩. В классическом подходе считается, что имена атрибутов уникальны, а значения представляют собой строки, поэтому каждая аннотация может содержать только одно значение каждого атрибута. При практической реализации часто требуется представлять множественные атрибуты, в этом случае пользуются одним из двух способов:

- создание нужного количества аннотаций, содержащих только атомарные значения атрибутов;

- использование некоторого способа кодирования информации для случая, когда в атрибуте требуется сохранить составное сложное значение (например, введение специального символа-разделителя).

Надо заметить, что разные модули системы могут использовать как один, так и другой способ. Второй способ является более универсальным, поскольку модули, поддерживающие его, также смогут проанализировать аннотации, созданные модулями, поддерживающими первый способ (обратной совместимости нет). При практической реализации мы следовали в основном второму способу.

Проблемы при использовании первого способа можно проиллюстрировать следующим примером. Пусть аннотация α обладает множеством атрибутов

$$A = \{a_1, a_2, \ldots, a_k\},$$

для каждого из которых, в свою очередь, определен набор из l_i значений

$$V_i = \langle v_{i\,1}, v_{i\,2}, \ldots, v_{i\,l_i} \rangle .$$

Тогда при классическом подходе к выражению вариативности каждая аннотация α должна быть преобразована в

$$N_\alpha = \prod_{i=1\ldots k} l_i$$

аннотаций с атомарными значениями атрибутов. Очевидно, что такой подход малоприменим для целого ряда классов аннотаций.

2.2. Правила и встроенные средства. Система применения правил также занимается различными видами лингвистического анализа. Она дает возможность быстро выполнять разработку новых уровней анализа. Кроме того, правила придают большую гибкость системе благодаря тому, что модификация правил выполняется существенно легче, чем модификация кода лингвистического процессора. Однако вычислительная эффективность анализа, построенного на правилах, ниже, чем у кода лингвистического процессора. Поэтому правила применять не всегда целесообразно.

Разделение функций между лингвистическим процессором и системой правил производится по критерию гибкость/эффективность. Лингвистический процессор не обладает большой гибкостью, но имеет высокую производительность кода. Система правил более медлительна, но легко поддается модификации.

Система сборки фрейма осуществляет поиск в тексте аннотаций определенного вида, которые связаны с целевой информацией. Так как фрейм — это многослотовая структура, нужно уметь различать целевую информацию, относящуюся к одному фрейму и к разным. Эту функцию осуществляет подсистема сборки фрейма.

3. Архитектура систем извлечения информации

Несмотря на то, что системы извлечения информации могут строиться для выполнения различных задач, подчас сильно отличающихся друг от друга, существует компоненты, которые можно выделить практически в каждой системе.

В состав почти каждой системы извлечения информации входят четыре основных компонента (Рис. 1), а именно: компонент разбиения на лексемы, некоторый тип лексического или морфологического анализа, синтаксический анализ (микро- и макроуровень), модуль извлечения информации и модуль для анализа на уровне конкретной предметной области[4]. В зависимости от требований к конкретному программному продукту, в приведенную выше схему добавляют дополнительные модули анализа (специальная обработка составных слов; устранение омонимии; выделение составных типов, которое может также быть реализовано на языке правил извлечения информации; объединение частичных результатов).

Разбиение на слова при анализе европейских языков не является проблемой, поскольку слова отделяются друг от друга пробелом (или знаками препинания). Тем не менее, для обработки составных слов, аббревиатур, буквенно-цифровых комплексов и ряда других особых случаев требуются специфические алгоритмы. С границами предложений, как правило, тоже больших проблем не возникает. Однако при анализе таких языков, как японский или китайский, определение границ слова на основе орфографии невозможно. По этой причине системы извлечения информации, работающие с такими языками, должны быть дополнены модулем сегментирования текста на слова.

В некоторые системы наряду с обычными средствами лексического и морфологического анализа могут быть включены модули для определения и категоризации атрибутов частей речи, смысловых нагрузок слов, имен или других нетривиальных лексических единиц.

[4]В действительности, многие системы ограничиваются морфологическим анализом и применением правил.

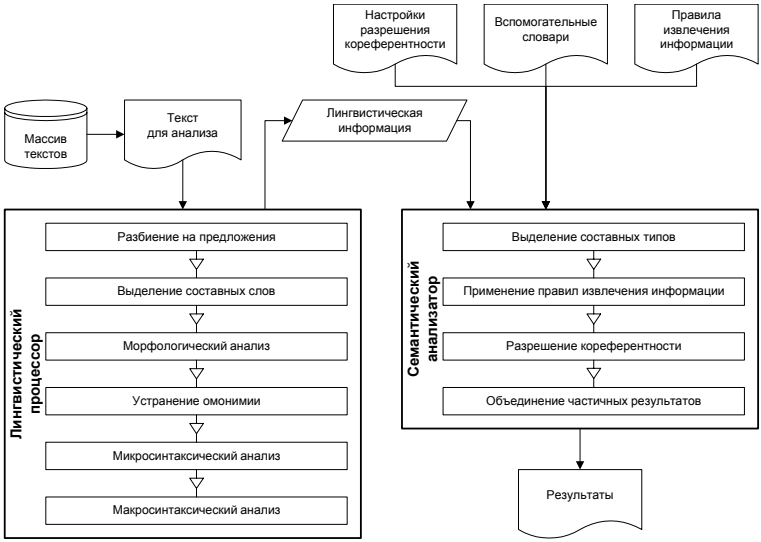

Рис. 1. Архитектура системы извлечения информации (для русского языка)

Для многих предметных областей элементарный синтаксический анализ (например, выделение именных групп) может быть достаточным для определения грамматической основы предложения, а также его основных частей, но в некоторых случаях может понадобиться расширенный или даже полный синтаксический анализ.

Существует различные методы первичного извлечения информации. В одних используются регулярные выражения «в чистом виде», в других методах пользуются простые правила на основе регулярных выражений, также существует подход с использованием специальных правил на основе целевых слотов и ограничений. Однако задача всех этих методов состоит в извлечении релевантной информации в локальном контексте, глобализация которого является задачей анализа на уровне предметной области.

Вполне возможно создание системы, которая не решает проблемы кореферентности и не объединяет целевые слоты, выделенные из различных предложений, в единый целевой фрейм. Однако во многих случаях включение модулей для решения этих проблем существенно повышает эффективность всей системы в целом.

Язык - орфография - пунктуация - морфология	Текст - структурированность - сложность - объем
Жанр - формализм - специфическая лексика	Задача - поиск объектов - свойства объектов - связи между объектами

Таблица 1. Факторы, влияющие на сложность системы

Подводя итоги вышесказанного, перечислим факторы (Табл. 1), определяющие, нуждается ли система в дополнительных модулях или можно проводить анализ с помощью основных компонент:

- *Специфика языка* анализируемого текста. В частности, ряд языков требует специальных модулей для разбиения текста на слова.
- *Жанр текста.* Извлечение информации из новостных статей требует технологии, отличной от анализа структурированных объявлений. Текст, написанный в свободном стиле, может содержать морфологические ошибки, неправильные грамматические конструкции, которые требуют специальной технологии анализа, которая не требуется при обработке новостных статей.
- *Особенности текста.* Очень длинный текст может потребовать специального модуля, определяющего приблизительно, какие части текста содержат релевантную информацию. Если же в тексте встречается информация в табличном виде, то необходимо отделить ее от остального текста и анализировать другими способами.
- *Задача анализа.* Задачи поиска релевантных объектов в тексте сравнительно просты. Если же вместе с объектами необходимо извлекать слова, которые подчеркивают их свойства, то, кроме определения простой грамматической основы предложения, необходим модуль фрагментации при синтаксическом анализе.

4. Инструментальные средства

На основе анализа требований рассмотрим набор инструментальных средств, необходимых системе извлечения информации, и наш подход к их реализации.

В ИЦИИ ИПС РАН в 2003 году была разработана программная система INEX [8]. Далее мы опишем наш взгляд на состав и функциональные характеристики инструментальных средств, основываясь на нынешнем состоянии программной системы и ближайших перспективах ее развития.

4.1. Стандартные средства. Для уменьшения объема работы по созданию и настройке конкретного приложения извлечения информации мы выделили набор *стандартных средств*, решающих наиболее типичные задачи обработки текста.

4.2. Организация библиотеки.

4.2.1. *Документы.* Одно из центральных понятий системы — *документ*, содержащий текст для анализа. В процессе обработки документ приобретает новые аннотации, выражающие информацию, полученную на разных этапах анализа. Документ обладает главной коллекцией аннотаций (см. раздел 4.2.2), в которой аннотации упорядочены сначала по возрастанию стартовой позиции, затем — по убыванию длины. Такой подход является наиболее естественным из-за того, что в процессе многоуровневой обработки текста преимущественно порождается набор вложенных аннотаций, описывающих представление текста на разных уровнях абстракции.

4.2.2. *Аннотации и коллекции аннотаций.* Информация, получаемая на разных этапах анализа текста, выражается при помощи *аннотаций*. Каждая аннотация описывает некоторую текстовую единицу с какой-либо точки зрения (с разной степенью абстракции). Например, слову в тексте может быть приписана морфологическая аннотация со множеством морфологических характеристик этого слова, описанных в атрибутах аннотации. Одному и тому же фрагменту текста может быть приписано несколько аннотаций, в том числе и принадлежащих одному классу. Таким образом выражается вариативность, присущая автоматическим системам обработки текста.

Очевидна необходимость оперировать наборами аннотаций. Как правило, новые аннотации строятся на основании не только текста, но и ранее выполненных фаз анализа. Таким образом при их построении

используются аннотации, порожденные анализаторами предшествующих фаз. В таких случаях часто возникают ситуации, когда нужно проанализировать все аннотации предыдущей фазы. Для удобства оперирования наборами аннотаций вводится понятие *коллекции аннотаций*.

Коллекция представляет собой контейнер векторного типа, то есть предоставляет доступ по индексу и набор стандартных функций, таких, как добавление, удаление и поиск аннотации.

В библиотеке различаются коллекции аннотаций двух видов: владеющие и невладеющие. Владеющие коллекции аннотаций при разрушении удаляют и входящие в них аннотации (отношение агрегирования). В отличие от них, невладеющие коллекции являются представлением для просмотра, аннотации не разрушаются при уничтожении коллекции. Невладеющие коллекции необходимы, например, при различного рода фильтрациях коллекций (см. раздел 4.2.4).

Для коллекции аннотаций определяется отношение порядка. Отношение порядка по умолчанию соответствует тому, которое описано в разделе 4.2.1. Возможна реализация других способов упорядочения аннотаций, наиболее соответствующих конкретной задаче анализа текста.

Каждая аннотация по умолчанию обладает одним атрибутом — ее численным уникальным (в пределах документа) идентификатором. При порождении новой аннотации (что возможно только для владеющих коллекций) выделяется новый идентификатор, для чего каждому документу сопоставлен генератор идентификаторов. При копировании аннотаций в невладеющую коллекцию (например, в результате фильтрации) идентификатор не меняется. Реализованы средства для поиска аннотации по идентификатору.

4.2.3. *Итераторы.* Итераторы представляют собой механизм последовательного просмотра контейнера. Они обеспечивают операции для перехода от одного элемента контейнера к другому. Использование итераторов позволяет абстрагироваться от конкретной реализации коллекции аннотаций и облегчает передачу параметров в функции. Различаются прямые и обратные итераторы. Для прямых итераторов операция перехода вперед осуществляет перебор от начала контейнера к его последнему элементу; для обратных — наоборот. Итераторы работают с учетом отношения порядка, определенного для коллекции. При реализации итератора можно использовать паттерн проектирования *Итератор*, описанный в [9].

4.2.4. *Фильтры.* Концепция *фильтров* заключается в том, чтобы на основании некоторого существующего контейнера построить новый контейнер, содержащий только те элементы, которые удовлетворяют некоторому предопределенному условию. Фильтры реализуются как функциональные объекты (функторы). Фильтры применяются к коллекции аннотаций. Реализованы следующие стандартные фильтры:

- по классам аннотаций;
- по границам аннотаций (строгая фильтрация, допускает аннотации, границы которых совпадают с заданным диапазоном);
- по границам аннотаций (нестрогая фильтрация, допускает аннотации, границы которых входят в заданный диапазон).

Перечисленный набор стандартных фильтров позволяет реализовать практически все виды фильтрации, необходимые для прикладного анализа текста.

4.2.5. *Прикладные задачи и анализаторы.* Сущность лингвистического анализа заключается в выявлении множества лингвистических атрибутов отдельных фрагментов текста. Среди атрибутов выделяют:

- графематические — определяют группы символов (слова), их форму, тип и язык;
- морфологические — определяют морфологические характеристики отдельного слова (род, падеж, число и т. д.);
- синтаксические — определяют связи и отношения между словами и группами слов;
- семантические — определяют смысл слов в данном контексте.

При реализации прикладной задачи обработки текста выделение каждого вида атрибутов выполняется отдельными модулями — *анализаторами.* Выполнение определенного вида анализа называется *фазой обработки.* Выделение какой-либо из групп вышеперечисленных атрибутов (за исключением простейшей графематики) невозможно (или нерационально) выполнять за один проход. Возможен лишь подход постепенного приближения к результату достаточно высокого качества. Так, например, тесно связаны между собой фазы морфологического и синтаксического анализа. По результатам морфологического анализа появляется довольно много морфологических

омонимов. Частичное разрешение омонимии возможно посредством синтаксического анализа. Качество синтаксического анализа находится в обратной зависимости от количества вариантов морфологического разбора. Таким образом, сложные комбинации выполнения фаз анализа морфологии и синтаксиса могут дать существенно более высокий результат, чем просто последовательное их выполнение. Последний (семантический) вид анализа тесно связан с предметной и проблемной областями, и суть его сильно зависит от задач, решаемых системой, использующей лингвистический анализатор. Поэтому здесь невозможно заранее реализовать весь необходимый анализ.

Для повышения гибкости системы и легкости ее настройки на новые предметные области реализована поддержка многофазовой обработки текста. Пользователи библиотеки могут применять в своих проектах встроенные анализаторы, а также разрабатывать свои и использовать их вместо или в дополнение к имеющимся. При этом пользователю предоставляется возможность выбирать анализаторы, которые необходимо применить при решении конкретной задачи, а также конфигурировать порядок их выполнения.

4.2.6. *Представления.* Не во всех случаях аннотации являются органичным и очевидным способом представления лингвистической информации. В частности, аннотации не позволяют в явном виде указывать связи между фрагментами текста, что необходимо для синтаксического или семантического представления текста при использовании модели дерева зависимостей.

Конечно, можно расширить базовую модель и представлять связи между фрагментами текста как особый объект — ссылку на две или более аннотаций, обладающий набором атрибутов. Но внесение существенных изменений в базовую модель аннотаций — тоже не лучшее решение, так как оно потребует поддержки новых объектов со стороны всех элементов системы, а также повлечет за собой изменения в языке правил. Если же отказаться от изменений в языке, то введение новой сущности окажется почти бессмысленным.

Другим способом решения этой проблемы может быть использование в аннотациях атрибутов особого вида, которые и будут задавать структурные отношения.

Для решения этой проблемы мы предлагаем комбинацию первого и второго подходов: использование *представлений* — функциональных объектов, осуществляющих «прозрачный» переход между

моделью аннотаций и естественным представлением для специфических лингвистических сущностей (и обратно). При этом данные попрежнему хранятся в атрибутах аннотаций (как подразумевается базовой моделью и предлагается во втором подходе). Для удобства программирования аннотации преобразуются объектом–представлением в нужную форму (например, дерево зависимостей), после чего клиентское приложение работает только с этим представлением. Когда трансформация закончена, представление может быть преобразовано обратно в аннотации.

4.2.7. *Фреймы результатов.* Результаты извлечения информации сохраняются в виде *фреймов* — древовидных структур с именованными узлами. Каждый узел может хранить значение текстового типа. Фрейм определяется при постановке задачи извлечения информации, создается *прототип фрейма* — фрейм с незаполненными полями значений. В дальнейшем, при обнаружении релевантной информации, происходит клонирование прототипа и заполнение значений. Все фреймы объединены в коллекцию, что позволяет производить уточнение результатов по итогам извлечения информации, например, объединить два или более частично заполненных фреймов в один, содержащий более полную информацию.

4.2.8. *Поддержка ввода-вывода.* В целях облегчения разработки и тестирования компонентов системы обработки текста предлагается ряд средств ввода–вывода информации. Большинство компонентов системы поддерживают сохранение в стандартные потоки (возможно, файловые) для изучения и отладки. Для некоторых элементов реализована сериализация и десериализация в бинарном виде, другие поддерживают вывод в удобочитаемый формат (HTML, текстовые файлы). Бинарная сериализация реализована для тех компонентов, которые с высокой степенью вероятности будут передаваться между отдельными исполнителями при реализации параллельной обработки, в частности, для результирующих фреймов. Вывод в HTML присутствует для коллекций аннотаций и фреймов результатов — для облегчения отладки при разработке. Также реализован текстовый ввод-вывод для коллекций аннотаций с целью организации автоматического тестирования.

4.2.9. *Конфигурирование.* Целью инструментальных средств является облегчение разработки программных средств извлечения информации из текстов, поэтому прикладная задача анализа текста

должна быть легко настраиваема. В нашем подходе настройка выражается в выделении анализаторов, каждый из которых выполняет строго определенную задачу обработки текста. Возможно создание средств, поддерживающих конфигурирование посредством специальных файлов (в простейшем случае, например, задающих последовательность фаз применения правил).

4.3. Правила извлечения информации.

4.3.1. *Характеристика языка описания правил извлечения информации.* Язык описания правил позволяет задать последовательность фаз, каждая из которых состоит из множества правил в виде «образец→действие». Фазы выполняются последовательно и функционально эквивалентны каскаду конечных автоматов, работающих на множестве аннотаций. Аннотации, полученные на ранних фазах, могут использоваться правилами из последующих фаз. Левая часть правила содержит образец для сопоставления. В образце могут использоваться следующие операторы:

- `|` — логическое «ИЛИ»;
- `!` — логическое «НЕ»;
- `?` — 0 или 1 соответствие;
- `*` — 0 или более соответствий;
- `+` — 1 или более соответствий.

Правая часть правила содержит операторы работы с аннотациями. При работе с аннотациями в правой части правила можно ссылаться на фрагменты образца в том случае, если последние сопровождались метками.

В левой части правила различают простые и составные условия. Простое условие описывает ограничения, накладываемые на одну аннотацию, и представляет собой конъюнкцию высказываний об атрибутах и их значениях для этой аннотации. Простые условия заключаются в фигурные скобки. Составное условие может включать в себя одно или несколько простых условий. Составные условия заключаются в круглые скобки. К составным условиям могут применяться операторы алгебры Клини (`?`, `*`, `+`). Составное условие может сопровождаться меткой для последующей ссылки на него из правой части правила. Метки представляют собой идентификаторы, перед которыми стоит двоеточие (`:`).

Последовательность простых или составных условий выражает последовательную конъюнкцию условий. Дизъюнкция условий выражается при помощи оператора `|` («ИЛИ»).

При сопоставлении строк в левой части правила можно использовать следующие операторы:

- `==` — сопоставление строки с учетом регистра;
- `=\` — сопоставление строки без учета регистра;
- `=|` — сопоставление строки с учетом морфологии (сопоставление канонических форм).

Для указания специальных символов в строках применяется символ `\`, то есть для указания, например, кавычек применяется конструкция `\"`.

Для облегчения разработки правил в языке описания правил существует механизм макроопределений, которые используются для сокращения объема исходного текста правил. В макроопределения выносятся фрагменты правил (из левой или правой части), которые повторяются в одном или нескольких правилах одной фазы.

4.3.2. *Общая схема применения правил извлечения информации.* Для применения правил извлечения информации используется *интерпретатор правил* (механизм извлечения информации) — модель, которая получает на вход аннотированный текст и порождает новые аннотации, если текст удовлетворяет заданным в ней правилам. Описания правил компилируются в модель путем построения обобщенной диаграммы переходов (конечного автомата). Такой автомат может быть детерминированным или недетерминированным (недетерминированный автомат может иметь более одного перехода из состояния при одном и том же входном элементе). Детерминированные конечные автоматы приводят к более быстрому распознаванию, однако они обычно больше по размеру, чем функционально эквивалентные им недетерминированные.

Для сравнения в Таблице 2 приводятся данные о памяти и времени, необходимых в наихудшем случае при работе с входным текстом x и языком, определяемым правилом r, с использованием распознавателей на базе детерминированных и недетерминированных конечных автоматов.

Автомат	Память	Время
НКА	$O(\|r\|)$	$O(\|r\| \times \|x\|)$
ДКА	$O\left(2^{\|r\|}\right)$	$O(\|x\|)$

Таблица 2. Распределение памяти и скорости работы для НКА и ДКА

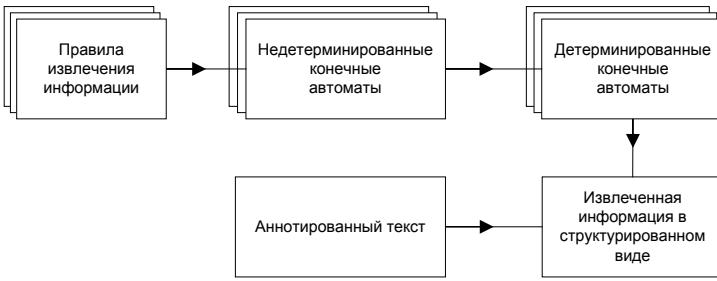

Рис. 2. Общая схема алгоритма применения правил

Для алгоритма применения правила извлечения информации существенна скорость работы, поэтому НКА, построенный по описанию правила, преобразуется в ДКА с использованием алгоритма построения подмножества. На Рис. 2 приведена общая схема алгоритма применения правил.

После преобразования правил в ДКА осуществляется применение правил и построение новых аннотаций. Применение правил идет по фазам, на каждой следующей фазе доступны аннотации, порожденные на предыдущих, если они были записаны в качестве входных параметров в спецификации фазы, а также аннотации по умолчанию.

Возможны следующие схемы применения правил:

(1) Применение каждого правила, начиная с очередной аннотации. Если правило срабатывает, то оно «поглощает» весь контекст, описанный в левой части. Далее рассмотрение продолжается с места, до которого дошло сработавшее правило. В случае конфликта правил определение порождаемых аннотаций происходит в соответствии с режимом применения правил.

(2) Применение поочередно каждого правила, начиная с каждой аннотации. В этом случае не происходит «поглощения», но усложняется процесс определения порождаемых аннотаций. Согласно данной схеме, каждое правило обнаруживает все возможные соответствия образцу из левой части для исследуемого фрагмента текста (по умолчанию — предложения).

В целях повышения гибкости инструментальных средств извлечения информации можно сделать выбор схемы применения правил одним из управляющих параметров целевой системы в целом, или же определять схему для каждой фазы или целевого приложения, использующего язык правил извлечения информации. Так, для проведения частичного синтаксического анализа предпочтительнее вторая схема, с сохранением всех полученных аннотаций и последующим анализом графовой структуры предложения для того, чтобы выбрать наиболее корректное синтаксическое представление предложения. В то же время для извлечения целевой информации или определения семантических классов больше подходит первая схема, так как она (при продуманном разбиении правил по фазам и расстановке приоритетов правил) позволяет избежать многих конфликтов между аннотациями и дает больший контроль над системой правил.

После выполнения всех правил фазы и отбора нужных аннотаций согласно режиму применения или в результате работы дополнительных алгоритмов происходит пополнение набора аннотаций фрагмента текста. Дополнительные алгоритмы могут потребоваться, например, для синтаксического анализатора — нельзя описать однозначный режим применения правил, который обладал бы достаточной степенью повторного использования.

Результатом работы прикладной системы является текст, сопровождаемый набором аннотаций, полученных в результате применения всех этапов обработки текста. Аннотации могут порождаться в результате многофазового применения правил или же специализированными алгоритмами.

5. Перспективы развития

Инструментальные средства предоставляют существенную базу для построения систем извлечения информации из текстов, но можно отметить ряд расширений, которые позволили бы еще упростить разработку конечных приложений.

(1) Стандартные средства конфигурирования прикладной задачи извлечения данных из текста.
(2) Стандартные средства построения фрейма предметной области.
(3) Расширение языка описания правил извлечения информации (поддержка логических ограничений; управление уровнем, на котором применяются правила; возможность ссылаться не только на фрагмент текста, но и на конкретную аннотацию).
(4) Средства для трассировки и отладки правил извлечения информации.
(5) Внесение ограничений на классы аннотаций, чтобы можно было указывать допустимость вложения, пересечения совпадения для аннотаций одного или разных классов.

Реальные прикладные задачи извлечения информации порождают в процессе работы значительный объем дополнительной информации о тексте, что неэффективно с точки зрения памяти. Существуют частные способы решения этой проблемы. Одним из них является введение средств *кэширования*. Поскольку часто приложения извлечения информации работают в однопроходном режиме или с незначительными отклонениями от него (например, для разрешения анафорических ссылок требуется просмотр нескольких предыдущих предложений), то можно хранить набор аннотаций не для всего текста, а только для его части. В случае, если происходит обращение к части текста, для которой лингвистическая информация не получена, можно осуществить повторный анализ «на лету». Естественно, для приложений, которым часто требуются обращения к существенно разным частям текста, это вызовет замедление работы, но для большинства приложений будет выигрыш по памяти без ухудшения производительности.

Заключение

Описанная в настоящей статье архитектура инструментальных средств позволяет существенно облегчить решение прикладных задач извлечения информации, что подтверждено экспериментами. Предложенная архитектура является расширяемой, что дает возможность вносить новые средства без нарушения функциональности существующих. Расширяемость заложена и в сам подход к описанию дополнительной информации о тексте, который обладает достаточной мощностью и, в то же время, не обладает главным недостатком фиксированных объектных моделей — хрупкостью и большим количеством внутренних зависимостей.

В процессе исследований мы также выделили перспективные направления развития архитектуры инструментальных средств систем извлечения информации.

Список литературы

[1] Appelt D. E., Israel D. *Introduction to information extraction technology* // IJCAI: tutorial, 1999. ↑(document)

[2] Хант Э. Искусственный интеллект. — М.: Мир, 1978. ↑(document)

[3] Nahm U. Y., Mooney R. J. *Mining Soft-Matching Rules from Textual Data* // IJCAI, 2001, с. 979–986. ↑(document)

[4] Кормалев Д. А., Куршев Е. П., Сулейманова Е. А., Трофимов И. В. *Извлечение информации из текста: методология и прикладные аспекты* // Труды Международных научно-технических конференций «Интеллектуальные системы» (IEEE AIS'03) и «Интеллектуальные САПР» (CAD-2003). — М.: Физматлит, 2003, с. 189–194. ↑(document)

[5] Кормалев Д. А., Куршев Е. П., Сулейманова Е. А., Трофимов И. В. *Приложения технологии извлечения информации из текстов: теория и практика* Прикладная и компьютерная математика. — Т. **2 №1**, 2003, с. 118–125. ↑(document)

[6] Кормалев Д. А., Куршев Е. П., Сулейманова Е. А., Трофимов И. В. *Извлечение данных из текста. Анализ ситуации ньюсмейкинга* // Труды восьмой национальной конференции по искусственному интеллекту с международным участием КИИ'2002. — Т. **1**. — М.: Физматлит, 2002. — ISBN 5–94052–056–1, с. 199–206. ↑1

[7] Grishman R. // TIPSTER Text Architecture Design. Version 3.1. — New York, NYU, 1998. ↑2.1

[8] Кормалев Д. А., Куршев Е. П., Осипов Г. С., Сулейманова Е. А., Трофимов И. В.: Препринт // Методы поиска и анализа информации. Автоматическое извлечение данных. — Переславль-Залесский, ИПС РАН, 2003. ↑4

[9] Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. Приемы объектно-ориентированного проектирования. Паттерны проектирования Библиотека программиста. — СПб.: Питер, 2001. ↑4.2.3

Исследовательский центр искусственного интеллекта ИПС РАН

Исследовательский центр медицинской информатики ИПС РАН

D. A. Kormalev, E. P. Kourshev, E. A. Suleimanova, I. V. Trofimov. *An Architecture for Information Extraction Tools.* (in russian.)

Abstract. The article outlines approaches to and architectural solutions for text processing and information extraction systems. General approaches to information extraction are described. Analysis of information extraction system needs and their typical structure is given. Finally, an architecture for information extraction tools is proposed, which is based on authors' experience.