В.В.Ланин

# Исследование корпусов с помощью среды GATE

Gate представляет собой программное обеспечение для обработки текстов на естественном языке. Программные продукты из семейства Gate отвечают различным потребностям пользователей.

Семейство Gate включает в себя:

• Gate Developer, интегрированную среду для разработки приложений по обработке текстов, включающую в себя встроенную систему извлечения информации и множество плагинов;

• Gate Cloud, средство для облачны вычислений;

• Gate Teamware, web-приложение для совместной работы над проектами;

• Gate Mimir, мультипарадигменный архив для поиска и индексирования данных;

• Gate Embedded, фреймворк, дающий доступ ко всем функциями Gate Developer.

Gate может использоваться как лингвистами, которые в данном случае являются конечными пользователями и применяют среду Gate Developer для решения своих задач с помощью имеющихся плагинов, так и программистами, которые огут использовать библиотеки Gate Embedded, чтобы написать собственный плагин или приложение с использованием функций, предоставляемых Gate.

Система обработки языка Gate (General Architecture for Text Engineering) содержит четыре основных компонента:

1. Applications: процессы обработки текста, применимые к отдельному документов или корпусу текстов.

2. Лингвистические ресурсы: документы, корпуса и онтологии.

3. Процессинговые ресурсы: инструменты аннотирования для неразмеченного или не полностью размеченного текста (txt, html, sgml, rtf, xml pdf).

4. Datastores: сохранение документов/корпусов и процессов для дальнейшего использования.

До начала работы в программе пользователь имеет различную релевантную информацию в форме массива текстов и множества атрибутов. После обработки корпуса текстов пользователь получает маркированные атрибуты, которыми размечены текстовые фрагменты.

## ANNIE

ANNIE (A Nearly New IE System) – система извлечения информации, компонент Gate.

Компоненты ANNIE:

1. Tokenizer (токенизатор: слова, числа, пунктуация, символ, пробел);
2. Gazetteer (словарь);
3. NE Transducer (семантчиеский теггер);
4. OrthoMatcher (орфографическая кореференция);
5. POS Tagger (теггер по частям речи);
6. Sentence Splitter (сплиттер предложений);
7. Document Reset (для удаления аннотаций).

### Tokenizer

Этот компонент служит для разбиения документа на последовательность токенов (tokens) - элементарных лексем типа чисел, слов, знаков пунктуации (Рис. 1). Правила по преобразованию текста в последовательность лексем указаны в файлах с расширением .rules папки $GATE\_HOME \plugins\ANNIE\resources\tokeniser.

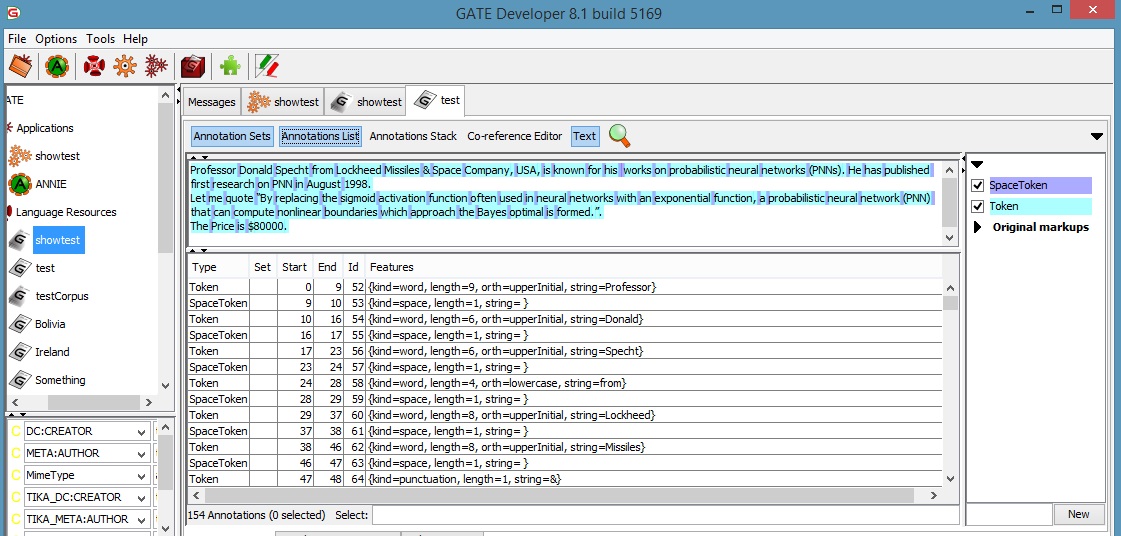


Рис. 1. Пример обработки документа ANNIE Tokenizer

Примеры правил (из файла DefaultTokeniser.rules)

1. "UPPERCASE\_LETTER" (LOWERCASE\_LETTER (LOWERCASE\_LETTER|DASH\_PUNCTUATION|FORMAT)\*)\* > Token;orth=upperInitial;kind=word;

Это правило ставит в соответствие последовательности символов, начинающихся с заглавной буквы и следующими за ней прописными буквами лексему (тип атрибута token) вида "слово" (kind=word), начинающегося с заглавной буквы (orth=upperInitial)

1. DECIMAL\_DIGIT\_NUMBER"+ >Token;kind=number;

Это правило ставит в соответствие последовательности цифр лексему (тип атрибута token) вида "число" (kind=number).

1. (SPACE\_SEPARATOR) >SpaceToken;kind=space;

Это правило ставит в соответствие пробельному символу лексему (тип атрибута SpaceToken) вида "пробел" (kind= space).

1. CONNECTOR\_PUNCTUATION|OTHER\_PUNCTUATION)>Token;kind=punctuation

Это правило ставит в соответствие символу пунктуации лексему (тип атрибута token) вида "пунктуация" (kind=punctuation).

### Gazetteer

Этот PR служит для выделения известных системе видов лексем (date, person, organization, company, currency и пр.). Результат работы - аннотирование известных лексем (или последовательности лексем) с помощью атрибута Lookup (Рис. 2).

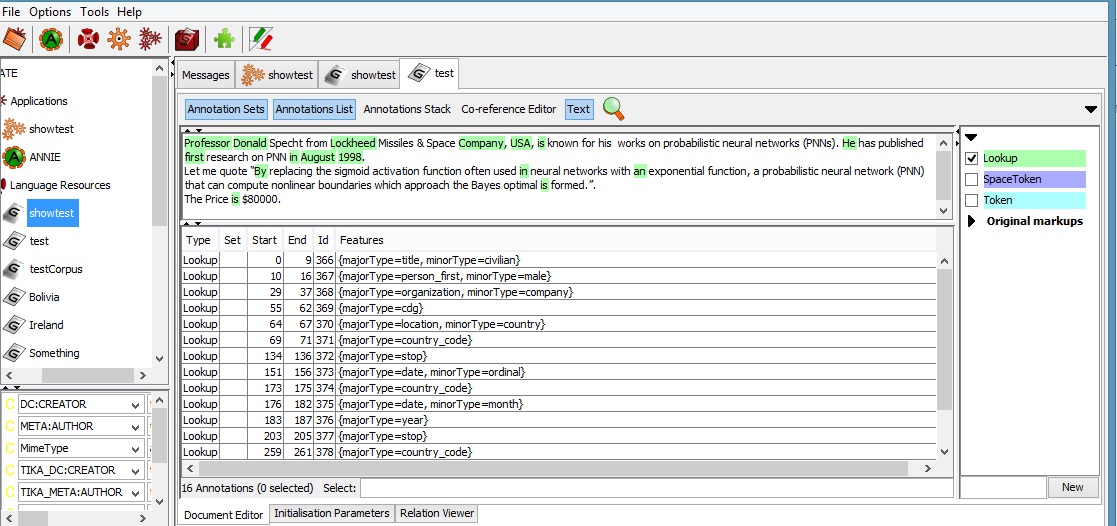


Рис. 2. Пример обработки документа ANNIE Gazetteer

Все известные виды лексем содержатся в файлах с расширением .lst в папке $GATE\_HOME \plugins\ANNIE\resources\gazetteer. Каждому виду лексем соответствует свой файл (city.lst, company.lst, country.lst, date.lst, year.lst, title.lst, person\_first.lst и др.).

Цель работы газетира - идентификация объектов в тексте на основе словаря этих объектов.

Словарь газетира содержится в обычном текстовом файле с расширением .lst и с одним элементом в каждой строке. Каждый словарь представляет собой набор имен, таких как наименования городов, организаций, дней недели и т.д.

Пример словаря денежных единиц:

Ecu   
European Currency Units   
FFr   
Fr   
German mark   
German marks   
New Taiwan dollar   
New Taiwan dollars   
NT dollar   
NT dollars

Заголовочный файл lists.def содержит перечень всех имен файлов со словарями. Они должны быть расположены в той же директории, что и заголовочный файл.

В заголовочном файле lists.def для каждого файла со словарем (.lst) указывается основной тип, опционально второстепенный тип, язык и тип аннотации через разделитель ":". В примере ниже первая колонка содержит имена файлов со списками, вторая колонка основной тип, третья - второстепенный тип, четвертая - язык, пятая - тип аннотации. Этот список извлекается при помощи конечного автомата. Любой текст, где найдено соответствие при помощи этого конечного автомата аннотируется с параметрами основного и второстепенного типа.

currency\_prefix.lst:currency\_unit:pre\_amount   
currency\_unit.lst:currency\_unit:post\_amount   
date.lst:date:specific\_date::Date   
day.lst:date:day   
monthen.lst:date:month:en   
monthde.lst:date:month:de   
season.lst:date:season

Основной и второстепенный тип добавляется в качестве параметров к аннотации Lookup из соответствующего словаря и соответствующей строки в словаре. Например, если для элемента из словаря газетира currency\_unit.lst найдено соответствие в некотором тексте, то обрабатывающий ресурс газетир создает для этого текста аннотацию Lookup с добавляет к аннотации параметры major="currency\_unit" и minor="post\_amount".

По умолчанию, ресурс ANNIE Gazetteer создает аннотации Lookup. Но пользователь может самостоятельно задать тип аннотации для всего словаря, тогда каждый элемент из этого словаря, найденный в тексте, будет аннотироваться заданным типом аннотации.

### NE Transducer

Этот компонент выделяет основные виды лексем (организации, даты, имена и пр.) в отдельные типы атрибутов (Рис. 3).

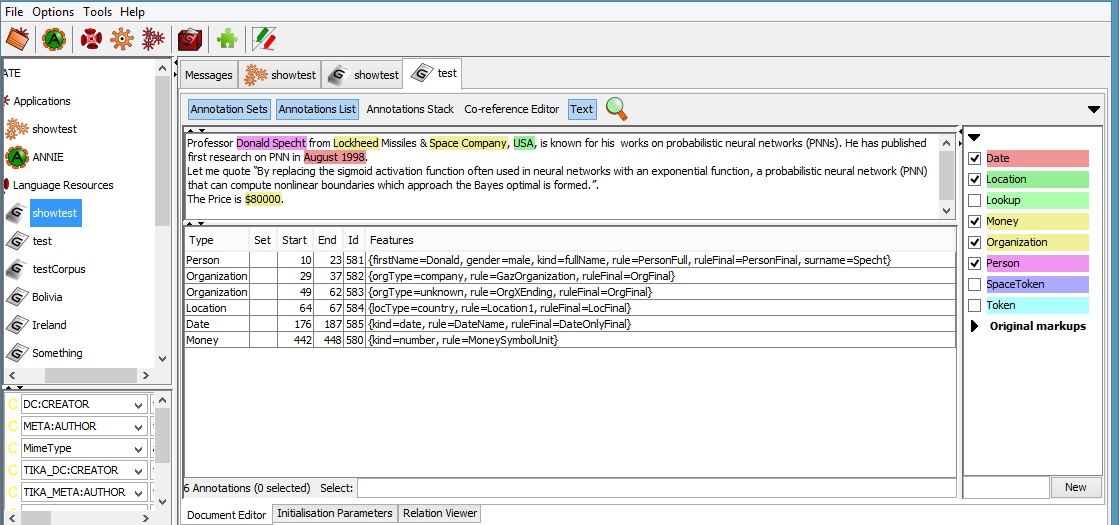


Рис. 3. Пример обработки документа ANNIE NE Transducer

### OrthoMatcher

Этот компонент распознаёт взаимосвязь между уже найденными семантическим теггером сущностями.

### POS Tagger

Этот компонент добавляет к каждой выделенной лексеме новое свойство category, которое описывает основные грамматические характеристики части речи (VB - verb, VBN - verb, past participle, IN - preposition, NNP - proper noun singular, полный список сокращений см. в приложении G к GATE user guide) (Рис. 4).

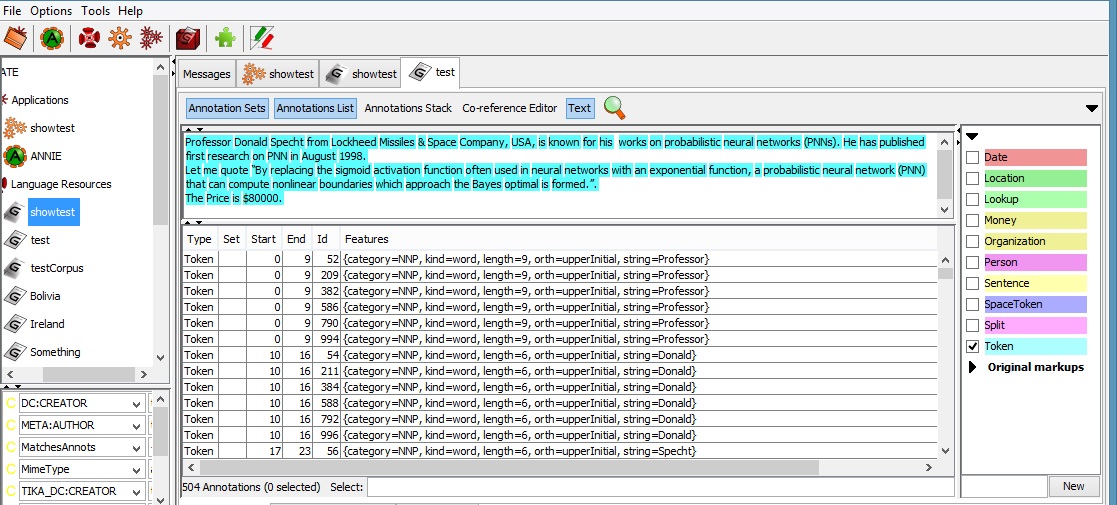


Рис. 4. Пример обработки документа ANNIE POS Tagger

### Sentence Splitter

Этот компонент служит для выделения предложений (Рис. 5).

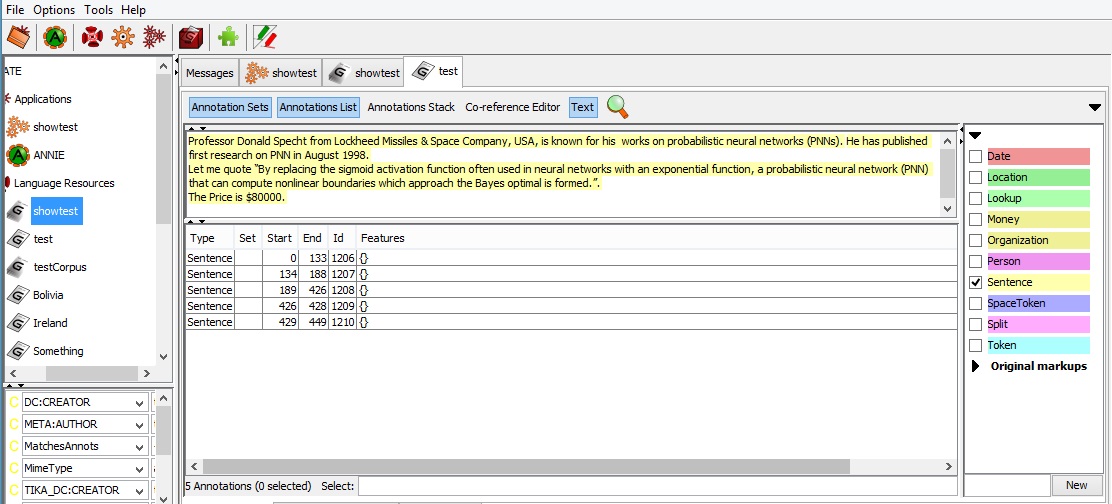


Рис. 5. Пример обработки документа ANNIE Sentence Splitter

### Document Reset

Этот компонент служит для удаления аннотаций из документа. При вызове с параметрами по умолчанию удаляются все аннотации. Обычно это PR применяется в качестве первого обрабатывающего ресурса приложения (application) Gate.

Доступны следующие основные параметры:

1. annotationTypes - типы аннотаций, которые должны быть удалены. Если этот параметр пустой, то удаляются все аннотации.
2. keepOriginalMarkupsAS - true, если требуется сохранить аннотации, которые были добавлены при загрузке документа (например, html-теги) и false, если требуется удалить все аннотации. Значение по умолчанию - true.

## Создание и изменение словаря

Словарь газетира может быть изменен при помощи любого текстового редактора или внутреннего редактора GATE, который запускается при двойном щелчке по газетиру в дереве ресурсов. Рекомендуется использовать редактор, который поддерживает кодировку UTF-8 (например редактор GATE Unicode).

Для создания словаря просто добавьте файл со словами в ту же директорию, что и имеющиеся словари, и внесите имя этого файла в заголовочный файл.

После всех модификаций словарей, проведенных при помощи внешнего редактора, реинициализируйте компонент газетир в GATE, если он уже был загружен, прежде чем запускать ваше приложение.

В случае использования редактора ANNIE Gazetteer (Рис. 6), откройте необходимый словарь, добавьте нужное слово и путём вызова контекстного меню ANNIE Gazetteer, сохраните и переинициализируйте газетир.

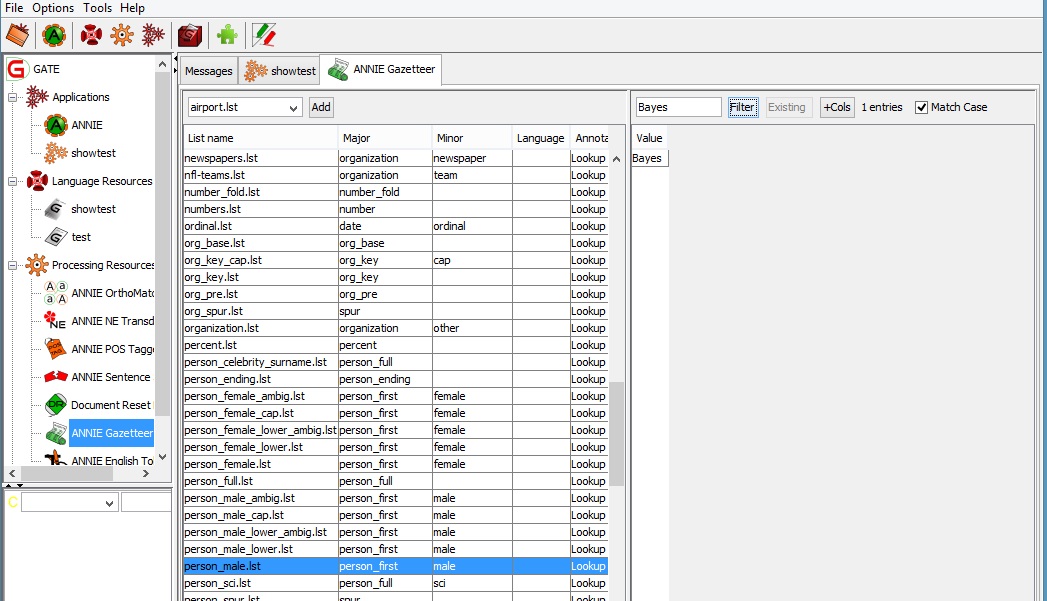


Рис. 6. Редактор ANNIE Gazetteer

## Создание обрабатывающего ресурса

Создадите новое приложение, вызвав контекстное меню компонента Application и выбрав: Create New Application->Corpus Pipeline. В появившемся окне находятся компоненты ANNIE, а также созданные вами ранее приложения, а также текстовые корпуса для обработки.

## Язык сценариев JAPE

Рассмотрим подробно синтаксис левой (LHS) и правой (RHS) части правила. Левая (LHS) часть содержит регулярное выражение для поиска необходимых фрагментов текста. Это выражение должно содержать только аннотации, перечисленные после строки "Input:" файла сценария (разделяемые пробелами). Далее перечислим некоторые примеры регулярных выражений:

• {Token} - любая лексема (фрагмент текста, выделенный с помощью аннотации с именем Token) • {Token, !Person} - любая лексема (фрагмент текста, выделенный с помощью аннотации с именем Token), которая не аннотирована как Person • {Token.kind==number} - лексема, у которой имеется атрибут kind со значением number. • Кроме атрибутов, которые можно увидеть у каждой аннотации в среде GATE Developer, доступен также ряд предопределенных атрибутов, основные из которых перечислены ниже

1. string - текстовое содержимое фрагмента текста, выделенное с использованием указанной аннотации, например, {Token.string=="января"} соответствует слову января. Если требуется выделить кавычки, то они экранируются обратным слешем

{Token.string=="\""}

1. length - длина (в символах) текстового содержимого фрагмента текста, выделенного с использованием указанной аннотации. Например, {Token.length<3} для выбора всех лексем, содержащих менее трех символов.

• Поддерживаются следующие операции сравнения: ==, !=, <, <=, >, >=. Кроме того, поддерживаются регулярные выражения - формат аналогичен регулярным выражениям, распознаваемым классом Pattern в языке Java (<http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html>). Поддерживаются 4 оператора

1. =~ соответствует методу Pattern.find() и возвращает истину, если левая часть оператора содержит подстроку, соответствующую регулярному выражению, указанному в правой части. Например, {Token.string=~"\d+"} соответствует любой лексеме, содержащей одну или несколько цифр, такой как 123, abc1def и пр.
2. !~ - отрицание оператора =~. Например, {Token.string!~"\d"} выделяет лексему, не содержащую цифр
3. ==~ соответствует методу Pattern.matches() и возвращает истину, если левая часть оператора в точности соответствует регулярному выражению, указанному в правой части. Например, {Token.string==~"\d+"} соответствует любой лексеме, состоящей только из цифр, такой как 123.
4. !=~ - отрицание оператора ==~. Например, {Token.string!=~"\d"} выделяет лексему, не совпадающую с одной из цифр, например, 12, abcd123efg.

• {Token.orth=="upperInitial", !Lookup.majorType == "person", !Lookup.minorType == "female" } - лексема, начинающаяся с заглавной буквы (значение атрибута orth равно upperInitial), но не аннотированная как персона женского пола (Lookup с атрибутами majorType, равным person, и minorType, равным female)

• Повторения реализуются символом \* (ноль и =ли несколько раз), + (один или насколько раз), ? (ноль или один раз). Например, ({Token.kind=number})+ соответствует последовательности из нескольких чисел. При этом выделяется максимально возможная последовательность. Если в тексте встретилась строка "abc 123 456 78 def", то будет выделена строка "123 456 78", но не ее части ("123 456" и др.).

• ({Token.kind=number})[3] соответствует ровно трем последовательно идущим числам.

• ({Token.kind=number})[2,5] соответствует последовательности лексем, содержащих от трех до 5 идущих подряд числел.

• ({Lookup.majorType == location} |{Lookup.majorType == country\_adj}) выделяет аннотированный как Lookup текст со значением атрибута majorType location ИЛИ country\_adj.

• Если требуется выделить несколько последовательно идущих аннотаций, то соответствующие им выражения указываются в LHS последовательно. Например, {Person} {Token.category=~"VB"} соответствует персона и следующий за ней глагол (лексема, атрибут category которой содержит VB)

• Можно поставить метку не только на всю LHS целиком, но и на ее часть (для выделения контекста). Например, следующее выражение позволяет использовать в правой части не только метку personAndVerb, но и метку verb.

( {Person} ({Token.category=~"VB"}):verb ):personAndVerb

Правая (RHS) часть может содержать:

1. Java-код для обработки документа.
2. Присвоение аннотации части текста, выделенной в регулярном выражении в LHS. Для этого используется следующий синтаксис: label.NewAnnotationName={attrName1=val1, attrName2=val2,...} Здесь label - это выделенная в части LHS метка (например, potentialdate в сценарии 1, см. предыдущий раздел), NewAnnotationName - имя вновь создаваемой аннотации, attrName1 - текстовое наименование атрибута аннотации, val1 - значение атрибута. Значение может содержать фиксированную константу или использовать значения атрибутов аннотаций, выделенных в LHS с помощью меток. Например, следующее правило присваивает тексту, отмеченному как Lookup со значением атрибута majorType, равным location, аннотацию MyLocation, значение атрибута type, который копируется из атрибута minorType, выделенного Lookup

Rule: LocationType

(

{Lookup.majorType == location}

):loc

-->

:loc.MyLocation = {rule = "LocationType", type = :loc.Lookup.minorType}

Кроме того, могут использоваться специальные атрибуты (string, length), отделенные от метки символом @. Например, следующее правило присваивает вновь создаваемой аннотации атрибут startDate со значением, совпадающим с текстом, выделенным с помощью метки startDate

(

(

{RussianDate}

):startDate

{Token.string=="-"}

)

:potentialDateRange

-->

:potentialDateRange.DateRange={startDate=:startDate@string }

## JAPE операторы

1. Ограничение на длину токенов

{Token.length == "3"} для GATE версии 4 и выше

{Token.length == 3} для GATE версии 5 и выше

1. Одновременное применение правил

{Token.length == "1", Token.kind == "word"} для GATE версии 4 и выше

{Token.length == "1", SpaceToken.kind == "control"} для GATE версии 5 и выше

1. Оператор отрицания

{Token.length == "1", !SpaceToken.kind == "control"}

ищет токен с длиной в 1 символ, который не имеет аннотации SpaceToken с атрибутом kind == "control"

1. Оператор покрытия и включения Contains and within operators

X включает Y

{X contains Y}

возвращает true, если аннотация типа X полностью включает в себя аннотацию типа Y.

X покрывается Y

{X within Y}

возвращает true, если аннотация типа X полностью покрывается аннотацией типа Y.

Можно задать правило так

{X contains {Y.foo==bar}}

и так

{X.f1 != "something", X contains {Y.foo=bar}}

и так

{Lookup.majorType == have, !Lookup within {VG.tense == PrePer}}

1. Оператор цепочек последовательных токенов

({Token})[2,5]

соответствует от 2 до 5 последовательных токенов.

## Условные обозначения JAPE

1. Копирование значений атрибутов из левой части (LHS) в правую часть (RHS) правила

Rule: Rule1

(

{Lookup.majorType == "location"}

):look

-->

:look.Location = { kind = :look.Lookup.minorType }

Правило работает только в том случае, если в документе присутствует аннотация Lookup только одного типа.

1. Использование найденных аннотаций в Java RHS

Такое правило:

-->

{

AnnotationSet tokAnnots = (AnnotationSet)bindings.get("tok");

if(tokAnnots != null && tokAnnots.size() != 0) {

// some Java code here

}

}

то есть вы можете использовать tokAnnots в Java коде и быть уверенными, что оно не пустое.

1. Использование дополнительных аннотаций

Rule: MeasureSpan

/\*

\* Matches loosely an interval.

\*/

({Token.kind == "word"})?:before

(

({Number}):amount1

((UNIT):unit1({Token.string == "."})?

)?

({Token.string == "-"}

|{Token.string == "±"}

|({Token.string == "+"}{Token.string == "/"}{Token.string == "-"})

):conj

({Number}):amount2

(UNIT)?:unit2

):span

-->

:amount1.Measurement = { type = "scalarValue"},

:unit1.Measurement = { type = "unit"},

:amount2.Measurement = { type = "scalarValue"},

:unit2.Measurement = { type = "unit"},

:span.Measurement = { type = "interval",

conj = :conj.Token.string,

before = :before.Token.string }

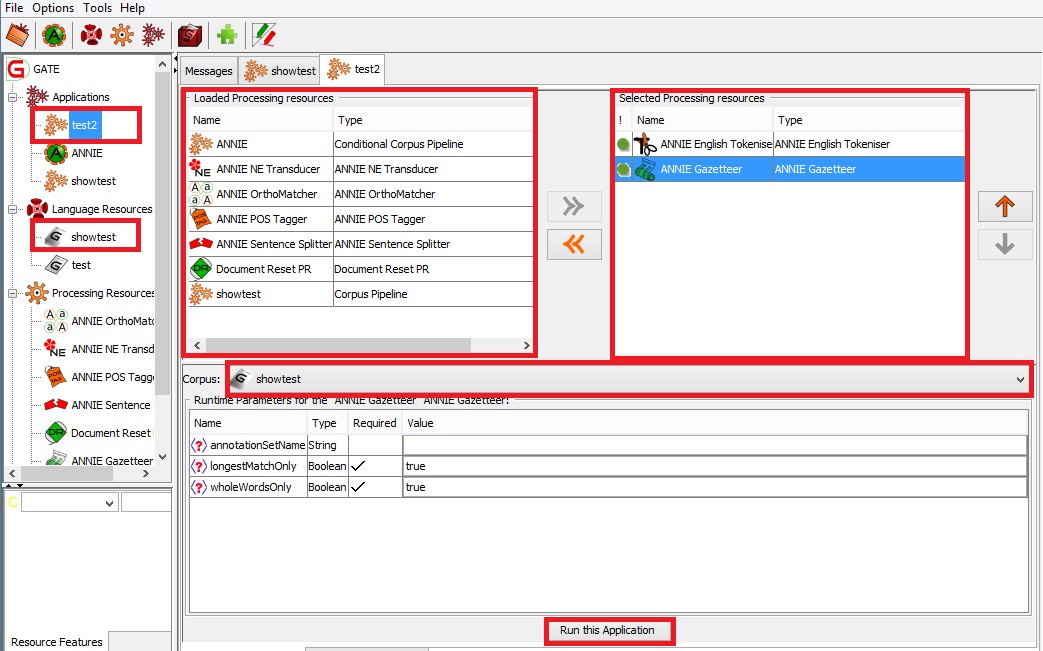


Рис. 7. Создание обрабатывающего ресурса

# Литература

1. Большакова Е.И. Лексико-синтаксические шаблоны в задачах автоматической обработки текстов / Н.В. Баева, Е.А. Бордаченкова, Н.Э. Васильева, С.С. Морозов // Компьютерная лингвистика и интеллектуальные технологии: Труды Международной конференции Диалог ‘2007 / Под ред. Л.Л. Иомдина, Н.И. Лауфер, А.С. Нариньяни, В.П. Селегея – М.: Изд-во РГГУ, 2007. – C. 70-75.
2. Jacobs P.S., Krupka G.R., Rau L.F. Lexico-semantic pattern matching as a companion to parsing in text understanding [Электронный ресурс] // URL: http://pdf.aminer.org/000/511/641/lexico\_semantic\_pattern\_matching\_as\_a\_companion\_to\_parsing\_in.pdf (дата обращения: 10.01.2015).
3. GATE JAPE Grammar Tutorial Version 1.0 [Электронный ресурс] // URL: https://gate.ac.uk/sale/thakker-jape-tutorial/GATE%20JAPE%20manual.pdf (дата обращения: 10.01.2015).
4. Большакова Е.И., Васильева Н.Э., Морозов С.С. Лексико-синтаксические шаблоны для автоматического анализа научно-технических текстов // Десятая Национальная конференция по искусственному интеллекту с международным участием КИИ-2006. Труды конференции в 3-х томах. Т. 2. – М.: Физматлит, 2006. – C.506-524.
5. Рабчевский Е.А. Автоматическое построение онтологий на основе лексико-синтаксических шаблонов для информационного поиска. // Труды 11-й Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» – RCDL’2009. – Петрозаводск, 2009. – С. 69-77.
6. Software, Tools, Lists, Resources [Электронный ресурс] // URL: http://www.uow.edu.au/~dlee/software.htm (дата обращения: 05.01.2015).
7. Linguistica | SourceForge.net [Электронный ресурс] // URL: http://sourceforge.net/projects/lxa/ (дата обращения: 08.01.2015).
8. Belter, C. W. A bibliometric analysis of NOAA’s office of ocean and exploration research // Scientometrics, 95(2).  с. 629–644.
9. GATE.ac.uk  OpenNLP Plugin [Электронный ресурс] // URL: https://gate.ac.uk/sale/tao/splitch23.html#sec:misc-creole:opennlp (дата посещения: 05.06.2015)
10. GATE.ac.uk  GATE Morphological Analyzer [Электронный ресурс] // URL: https://gate.ac.uk/releases/gate-6.0-build3764-ALL/doc/tao/splitch17.html#sec:parsers:morpher (дата посещения: 05.06.2015)