Author: Brian Davis

Organisation: Digitial Enterprise Research Institute, Galway

Version: 1.9

Last Revision: 11th March 2010

GATE Version: Release 4.0 **(July 12th 2007)**

1. Learning Outcomes

You will by the end of this tutorial:

- have basic knowledge of the GATE Deveploment Environment .
- know how to create a simple NLP application of your own in the GATE Development Environment.
- know how to create the same GATE application programatically.
- learn how to exploit IE in GATE
- have been exposed to JAPE for NE Extraction and Relation Extraction

Glossary

GATE –	General Architecture for Text Engineering
NLP –	Natural Language Processing
IE –	Information Extraction
JAPE –	Java Annotation Patterns Engine
NE –	Named Entities

Materials:
This document and source code to be provided either at the beginning of the Tutorial or in the tutorial folder accompanying this document.

2. About GATE

"GATE is an infrastructure for developing and deploying software components that process human language. GATE helps scientists and developers in three ways":

1. by specifiying an architecture, or organisational structure, for language processing software;
2. by providing a framework, or class library, that implements the architecture and can be used to embed language processing capabilities

in diverse applications;
3. by providing a development environment built on top of the framework made up of convenient graphical tools for developing components."

[1]

Recommended Reading:

[1]GATE User Guide See http://gate.ac.uk/sale/tao/split.html
[2] Software Architecture for Language Engineering
http://gate.ac.uk/sale/thesis/

If you have not already, you should go through the following material: of the GATE User Guide

Chapter 1, 6, 8

3. Practical

Task 1

Start Gate either using:

Use your platform specfic installer
See Gate User Guide Section 2.2.1

[1]http://gate.ac.uk/sale/tao/splitch2.html#x5-220002.2.1

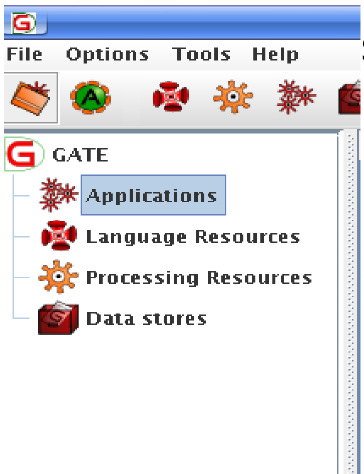
OR

```
cd YourGateHomeDir

bin/ant run   #(Linux or Unix)

bin/ant.bat run #Windows
```

In the USB stick (if provided)

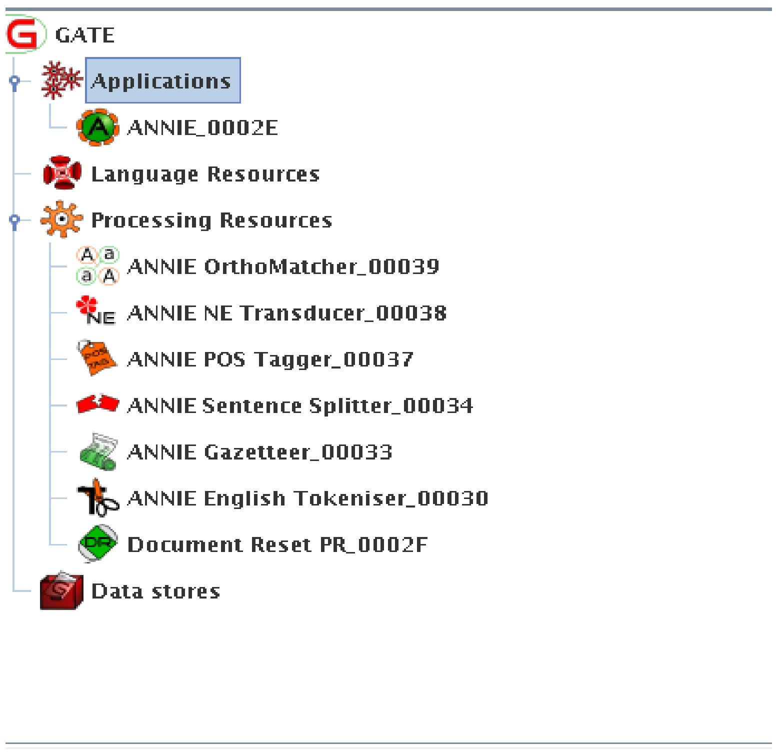
Task 2

Load ANNIE (A Nearly New Information Extraction System) with defaults.



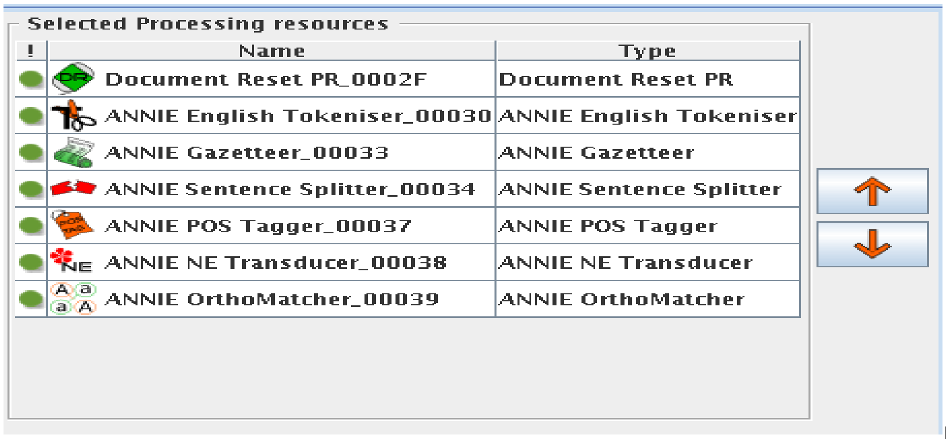
See Gate USER Guide Chapter 6 for further details, Click on and choose "Load ANNIE with defaults

The following will be loaded:



An instance of a Serial Analyser Controller will appear under "Applications" with the name "Annie_####". This type of Controller is used to manage a pipeline of NLP processing resources which will be run over a corpus. A corpus is a collection of documents. GATE consists of Processing Resources (PRs) – Programs or Algorithms which will do some sort processing on text i.e. Tokenising or dictionary lookup, parsing etc and Language Resources (LRs) – documents, Corpora, Ontologies – data essentially .The third component is Visual Resources (Vrs) but you do not need to worry about them here. All GATE Resources are realised as Java Beans. See Chapter 1 in the Gate User Guide for more details.

If you double click on the the "ANNIE" instance, the following will appear:


Selected Processing resources

!	Name	Type
●	Document Reset PR_0002F	Document Reset PR
●	ANNIE English Tokeniser_00030	ANNIE English Tokeniser
●	ANNIE Gazetteer_00033	ANNIE Gazetteer
●	ANNIE Sentence Splitter_00034	ANNIE Sentence Splitter
●	ANNIE POS Tagger_00037	ANNIE POS Tagger
●	ANNIE NE Transducer_00038	ANNIE NE Transducer
●	ANNIE OrthoMatcher_00039	ANNIE OrthoMatcher

Notice how the PRs you initially loaded when you double clicked on the "ANNIE with defaults" icon, where also automatically added to the pipeline in the appropriate order. Note that the order of you PRs is very important, for instance you cannot really split sentences until you have tokenised the document, this is much more subtle with respect to JAPE processing later. Briefly, the default PRs loaded with ANNIE are:

Document Reset PR
GATE keeps annotations and text seperately (StandOff Markup), in fact any GATE document will pass through a pipleine unaltered – in other words no annotation or parsing information is embedded in a document. For design reasons which are beyond the scope of this tutorial, GATE tends to employ weak referencing in the JVM, thus overiding garbage collection. Consequently you will need to explicity delete resources – mostly documents when using GATE programatically. GATE will also keep annotation sets derived from previous processing in memory. This has two consequences:

1)Old annotations residing in memory from a previous document may be taken into account when processing the next document.

2)The JVM heap will begin to fill up with old annotation sets generated from previous documents which can severely impact on performance.

Hence the purpose of the Document Reset PR is to clean out any previous annotations, essentially releasing the annotation set objects for garbage collection.

ANNIE English Tokeniser

This PR splits a text into annotations of type Token, taking into account spaces etc. In addition a JAPE transducer is used to conduct post processing in order to handle contractions in English i.e. Don't, Can't. **Only use the ANNIE English tokeniser if you a certain your text contains only English.** In addition, if your document contains junk "markup" attempt one way or another to clean it out of the text or ensure that it is well formed. Failing this, will cause the English tokeniser to choke on certain input, particularly at the post processing phase. If you cannot ensure this, use the GATE Unicode Tokeniser.

Always use the GATE Unicode Tokeniser for languages other than English.

ANNIE Gazetteer

This PR compiles wordlists into Finite State Machines in order to conduct NE(Named Entity) or Key Phrase look up. Output is annotations of type *LookUp*. Used for NE Recognition. An NE is a: Person, Location Date, Address **See http://en.wikipedia.org/wiki/Named_entity_recognition**

ANNIE Sentence Splitter

Assigns annotations of type *Split* to sentence boundaries in text based on punctuation and \t \r keys. It is implemented as cascade of JAPE transducers. Creates *Sentence* annotations.

ANNIE POS Tagger

The POS (Part of Speech) tagger assigns POS tags to Token annotations.
The (Hepple) tagger is a modified version of the Brill tagger. The list of tags used is given in Appendix D of the GATE User Guide[1]. The tagger uses a default lexicon and ruleset (the result of training on a large corpus taken from the Wall Street Journal). A POS tag is the assignment of linguistic category to a Token i.e. NNP or NNPS for Proper Nouns in singular or plural froms respectively.

ANNIE NE transducers

This PR will

(1) attempt to find unknown Named Entities based on extraction templates written in the JAPE language i.e. A Token with an NNP (proper Noun) POS tag followed by a Lookup annotation for Company suffixs like "Ltd or "GmBH" should be annotated as an NE of type Company or Organisation.
(2) act on Lookup annotations and additional local contextual linguistic information to output a final Semantic Type or NE. i.e. Lookups of type "title" followed by a "firstname" and "lastname" should be annotated entitely as an NE of type Person - "Mr John Smith"

A Brief Note on JAPE

JAPE – Java Annotation Patterns Engine

JAPE provides finite state transduction over annotations based on regular expressions. JAPE is based CPSL – Common Pattern Specification Language – See[1] Chapter 8

A JAPE grammar consists of a set of phases, each of which consists of a set of pattern/action rules. The phases run sequentially and constitute a cascade of finite state transducers over annotations. The left-hand-side (LHS) of the rules consist of an annotation pattern that may contain regular expression operators (e.g. *, ?, +). The right-hand-side (RHS) consists of annotation manipulation statements in the Java code. Annotations matched on the LHS of a rule may be referred to on the RHS by means of labels that are attached to pattern elements. Cascaded FSTs are shallow or partial parsers, examples inlcude FASTUS[3] and Jfrost within IBM LanguageWare[4]. They are often employed in IE beacuse they are robust and efficient. One could achieve the same result using a deep parser, however this would impact significantly on performance and robustness, also the majority of parsing information would be discarded in order to complete the same task in IE.

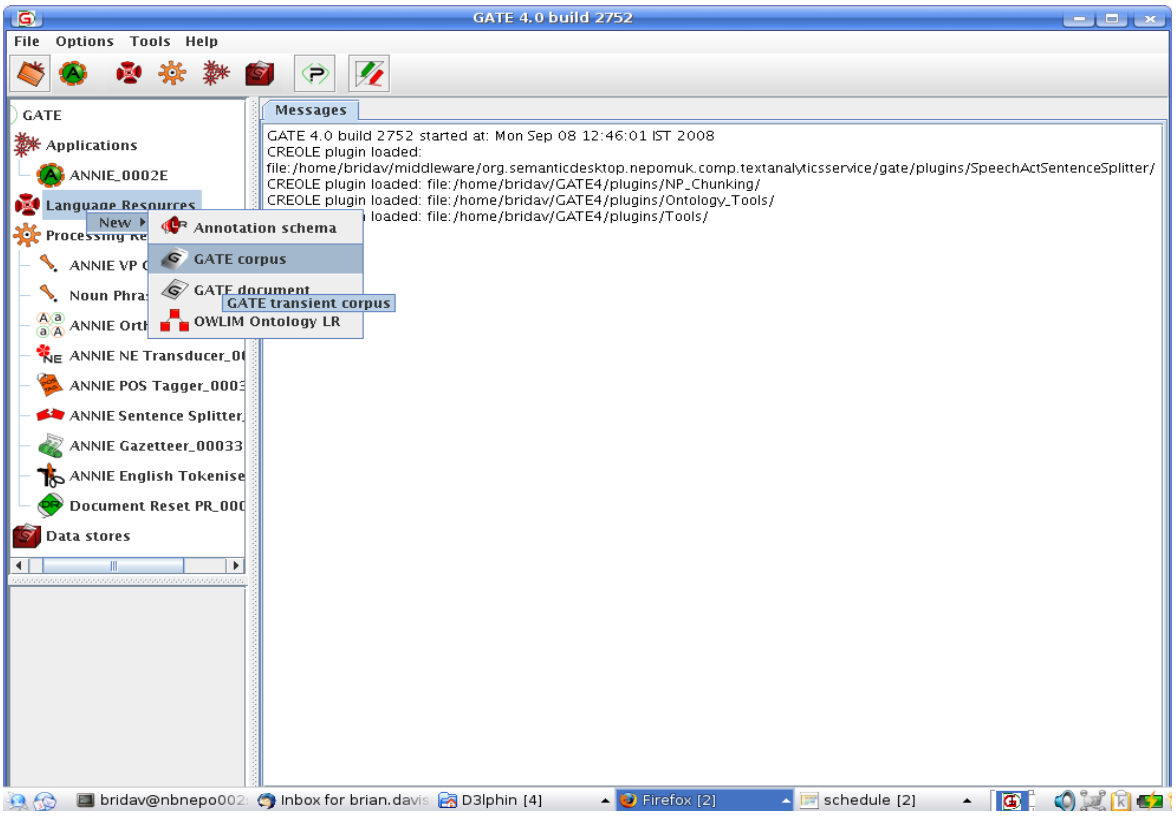
We will look at a JAPE in detail later. If you double click on the NE Transducer in the Resources Tree you can view a JAPE file which calls a collection of phases, similar to a `main` method in C or Java

Orthomatcher PR

The Orthomatcher module adds identity relations between named entities found by the NE Transducer, in order to perform coreference. It does not find new named entities as such, but it may assign a type to an unclassified proper name, using the type of a matching name.
The matching rules are only invoked if the names being compared are both of the same type, i.e. both already tagged as (say) organisations, or if one of them is classified as ‘unknown’. This prevents a previously classified name from being recategorised. An alias list exists also for Orthormatcher i.e. The PR would create an identity relation between the “Big Blue” and “IBM”. The alias list however most be manually constructed.

Task 3

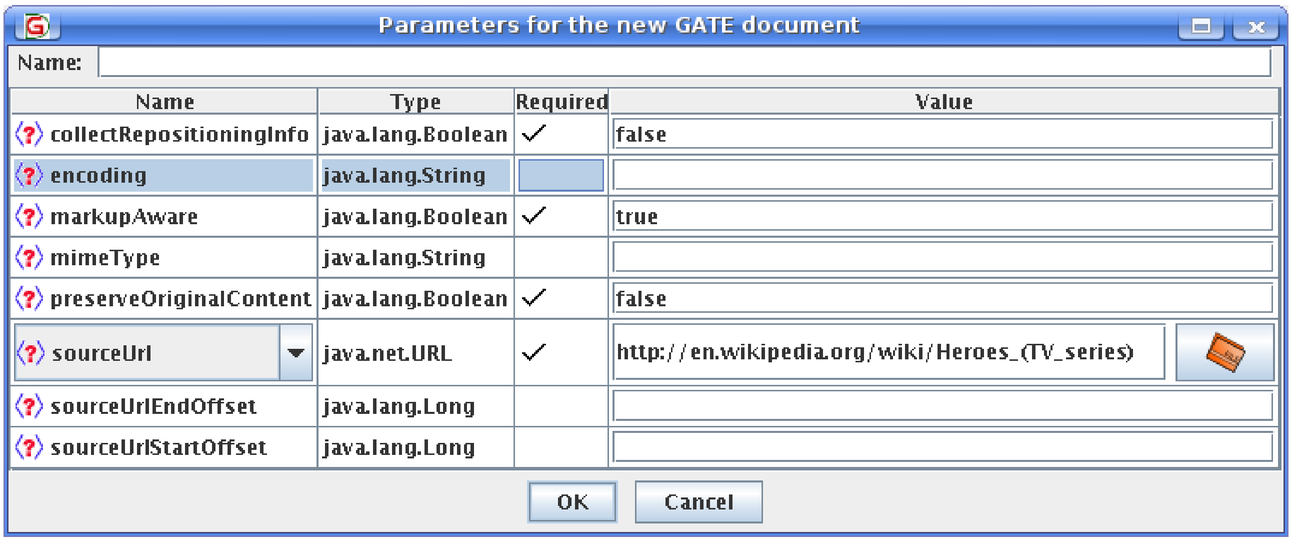
Run ANNIE over a document.



```
GATE 4.0 build 2752 started at: Mon Sep 08 12:46:01 IST 2008
CREOLE plugin loaded:
file:/home/bridav/middleware/org.semanticdesktop.nepomuk.comp.textanalyticsservice/gate/plugins/SpeechActSentenceSplitter/
CREOLE plugin loaded: file:/home/bridav/GATE4/plugins/NP_Chunking/
CREOLE plugin loaded: file:/home/bridav/GATE4/plugins/Ontology_Tools/
 loaded: file:/home/bridav/GATE4/plugins/Tools/
```

Right Click on Language Resources and click on GATE corpus and click OK

Repeat the same for GATE document however, when prompted for the source URL, either provide a URL to for example your favourite TV Program on the Web (if internet access is working or alternatively choose a file from your desktop or from the `documents` subfolder in the `tutorial` folder.



Name:

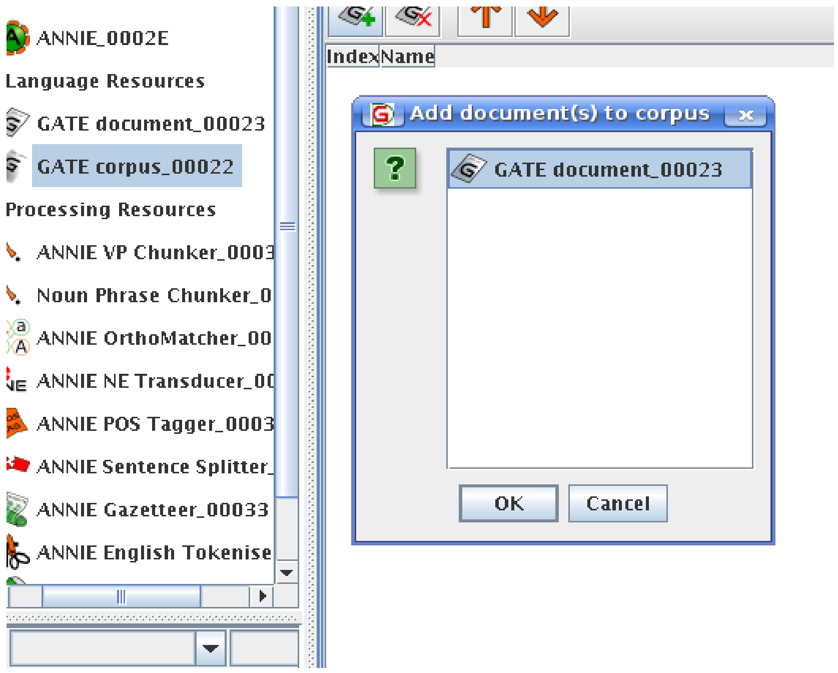
Name	Type	Required	Value
collectRepositioningInfo	java.lang.Boolean	✓	false
encoding	java.lang.String		
markupAware	java.lang.Boolean	✓	true
mimeType	java.lang.String		
preserveOriginalContent	java.lang.Boolean	✓	false
sourceUrl	java.net.URL	✓	http://en.wikipedia.org/wiki/Heroes_(TV_series)
sourceUrlEndOffset	java.lang.Long		
sourceUrlStartOffset	java.lang.Long		

GATE will handle most file formats:

Plain Text

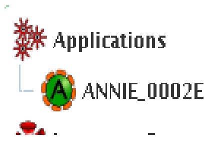
- HTML
- SGML
- XML
- RTF
- Email
- PDF (some documents)
- Microsoft Word (some documents)

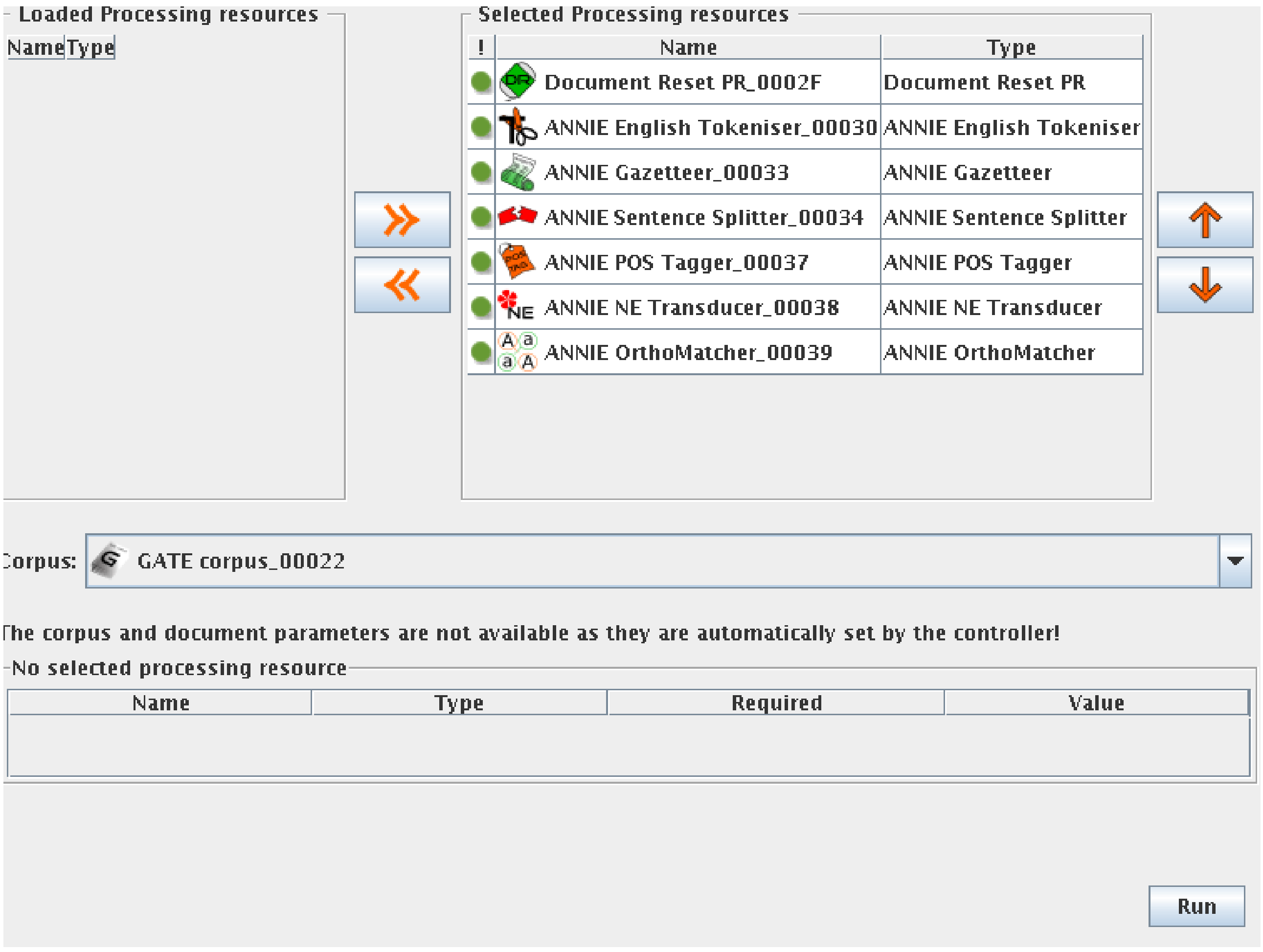
Recall the the application we are using is a Corpus pipeline. We are not processing a single document. If that were the case, we would have created a serial controller instead.



Hence we will need to add our document to our Corpus. Right Click on the Corpus Icon in the Resources and click on the following icon to add your document.

Double Click on the ANNIE application icon again .





Loaded Processing resources

Name	Type

Selected Processing resources

!	Name	Type
●	Document Reset PR_0002F	Document Reset PR
●	ANNIE English Tokeniser_00030	ANNIE English Tokeniser
●	ANNIE Gazetteer_00033	ANNIE Gazetteer
●	ANNIE Sentence Splitter_00034	ANNIE Sentence Splitter
●	ANNIE POS Tagger_00037	ANNIE POS Tagger
●	ANNIE NE Transducer_00038	ANNIE NE Transducer
●	ANNIE OrthoMatcher_00039	ANNIE OrthoMatcher

Corpus: GATE corpus_00022

The corpus and document parameters are not available as they are automatically set by the controller!

No selected processing resource

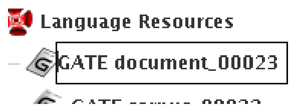
Name	Type	Required	Value

The ANNIE pipeline shown in the main viewing pane (see above) . If the corpus is not already loaded use the pulldown selector to find it. (To the right of **Corpus:** in the view pane)

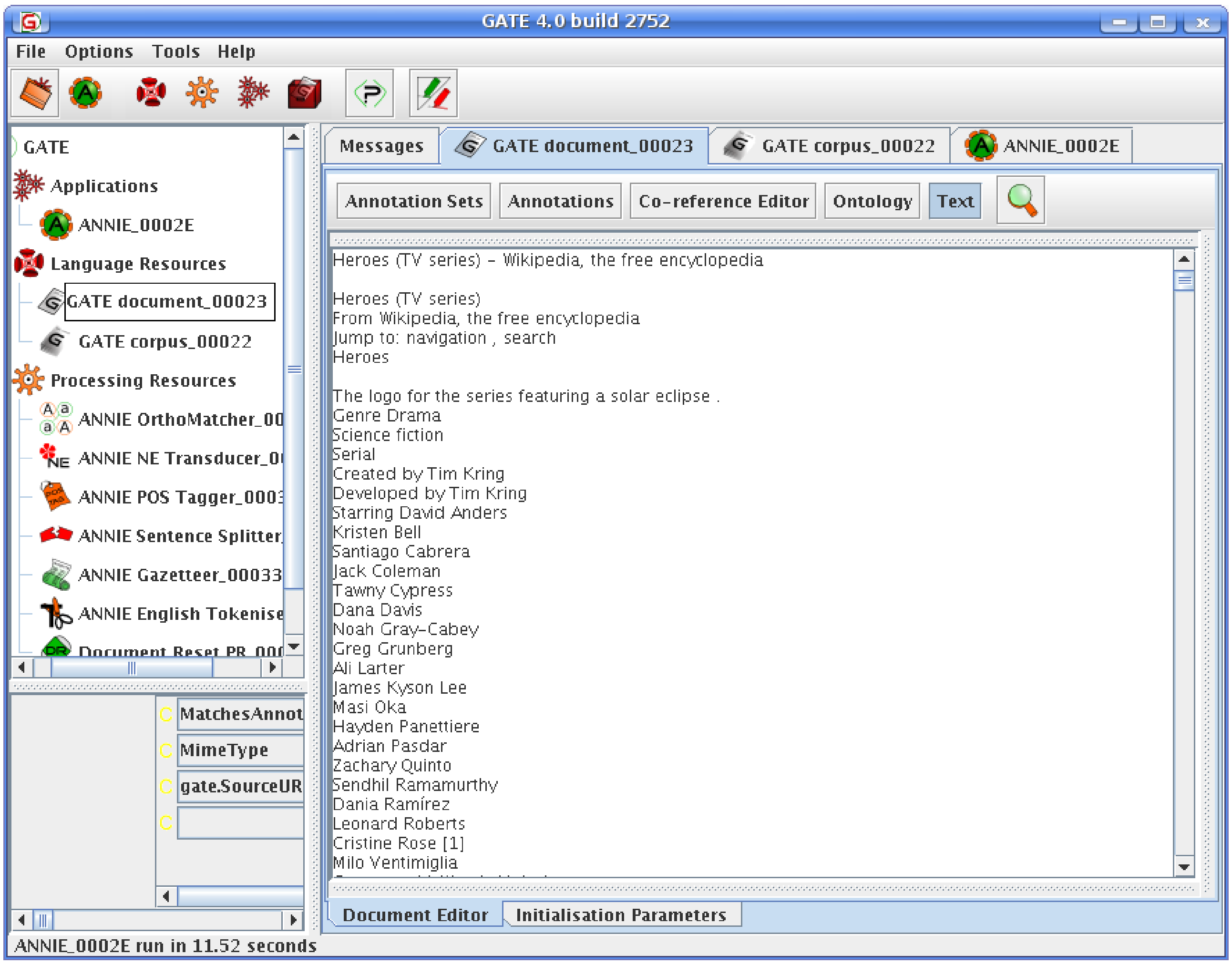
Run ANNIE by clicking the Run Button

Wait a few seconds :-)

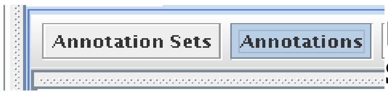
Now click on your GATE document in Language Resources tree. See icon below.



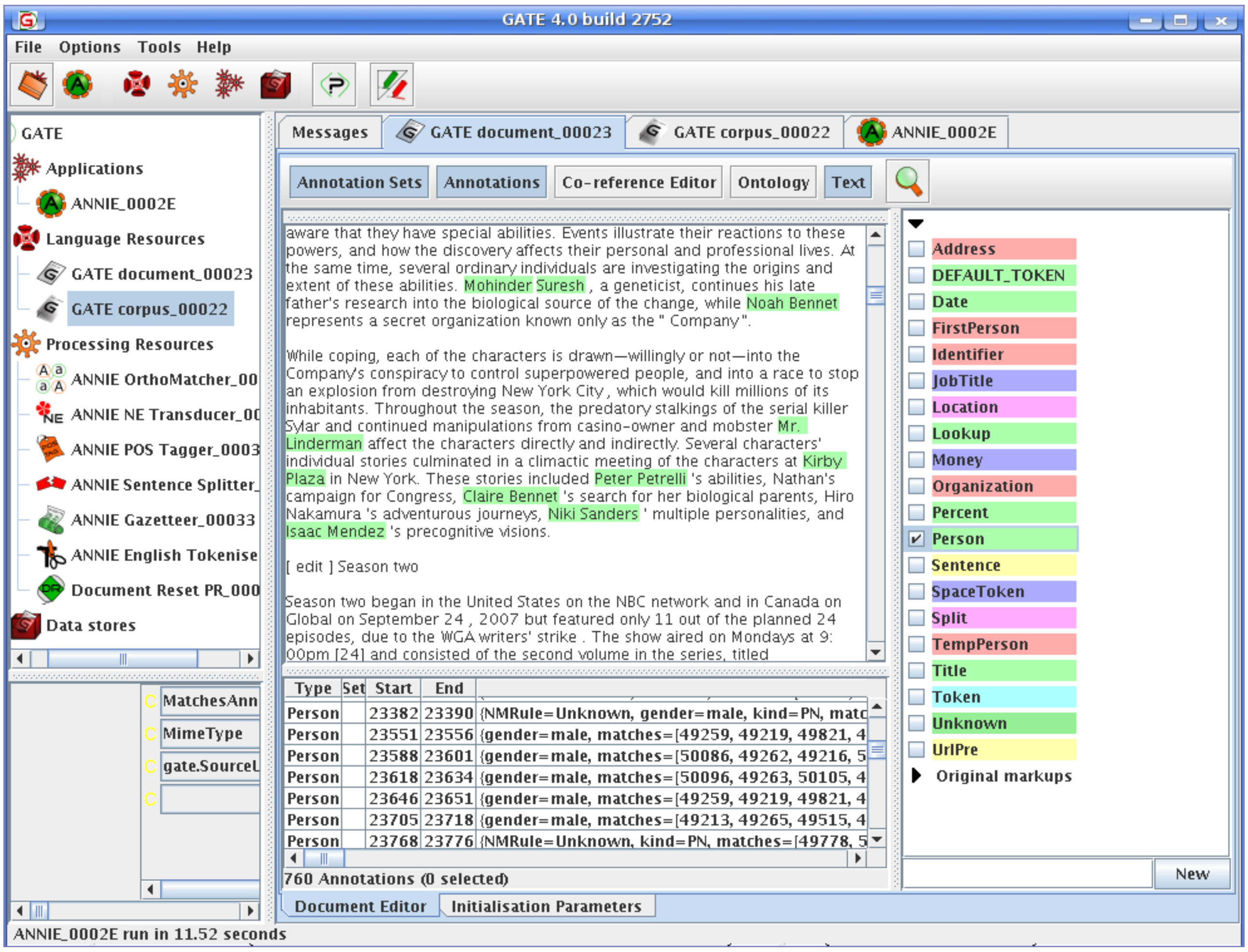
The document viewer should open as shown below.



```
Heroes (TV series) - Wikipedia, the free encyclopedia

Heroes (TV series)
From Wikipedia, the free encyclopedia
Jump to: navigation , search
Heroes

The logo for the series featuring a solar eclipse .
Genre Drama
Science fiction
Serial
Created by Tim Kring
Developed by Tim Kring
Starring David Anders
Kristen Bell
Santiago Cabrera
Jack Coleman
Tawny Cypress
Dana Davis
Noah Gray-Cabey
Greg Grunberg
Ali Larter
James Kyson Lee
Masi Oka
Hayden Panettiere
Adrian Pasdar
Zachary Quinto
Sendhil Ramamurthy
Dania Ramírez
Leonard Roberts
Cristine Rose [1]
Milo Ventimiglia
```



Double Click on both **Annotations and Annotation Sets** buttons in the documentviewer.



aware that they have special abilities. Events illustrate their reactions to these powers, and how the discovery affects their personal and professional lives. At the same time, several ordinary individuals are investigating the origins and extent of these abilities. Mohinder Suresh , a geneticist, continues his late father's research into the biological source of the change, while Noah Bennet represents a secret organization known only as the " Company".

While coping, each of the characters is drawn—willingly or not—into the Company's conspiracy to control superpowered people, and into a race to stop an explosion from destroying New York City , which would kill millions of its inhabitants. Throughout the season, the predatory stalkings of the serial killer Sylar and continued manipulations from casino-owner and mobster Mr. Linderman affect the characters directly and indirectly. Several characters' individual stories culminated in a climactic meeting of the characters at Kirby Plaza in New York. These stories included Peter Petrelli 's abilities, Nathan's campaign for Congress, Claire Bennet 's search for her biological parents, Hiro Nakamura 's adventurous journeys, Niki Sanders ' multiple personalities, and Isaac Mendez 's precognitive visions.

[edit] Season two

Season two began in the United States on the NBC network and in Canada on Global on September 24 , 2007 but featured only 11 out of the planned 24 episodes, due to the WGA writers' strike . The show aired on Mondays at 9:00pm [24] and consisted of the second volume in the series, titled

Type	Set	Start	End	
Person		23382	23390	{NMRule=Unknown, gender=male, kind=PN, matc
Person		23551	23556	{gender=male, matches=[49259, 49219, 49821, 4
Person		23588	23601	{gender=male, matches=[50086, 49262, 49216, 5
Person		23618	23634	{gender=male, matches=[50096, 49263, 50105, 4
Person		23646	23651	{gender=male, matches=[49259, 49219, 49821, 4
Person		23705	23718	{gender=male, matches=[49213, 49265, 49515, 4
Person		23768	23776	{NMRule=Unknown, kind=PN, matches=[49778, 5

760 Annotations (0 selected)

ANNIE_0002E run in 11.52 seconds

Annotation Sets will appear on the Right hand side and Annotations (and their features) will appear on the left hand side.

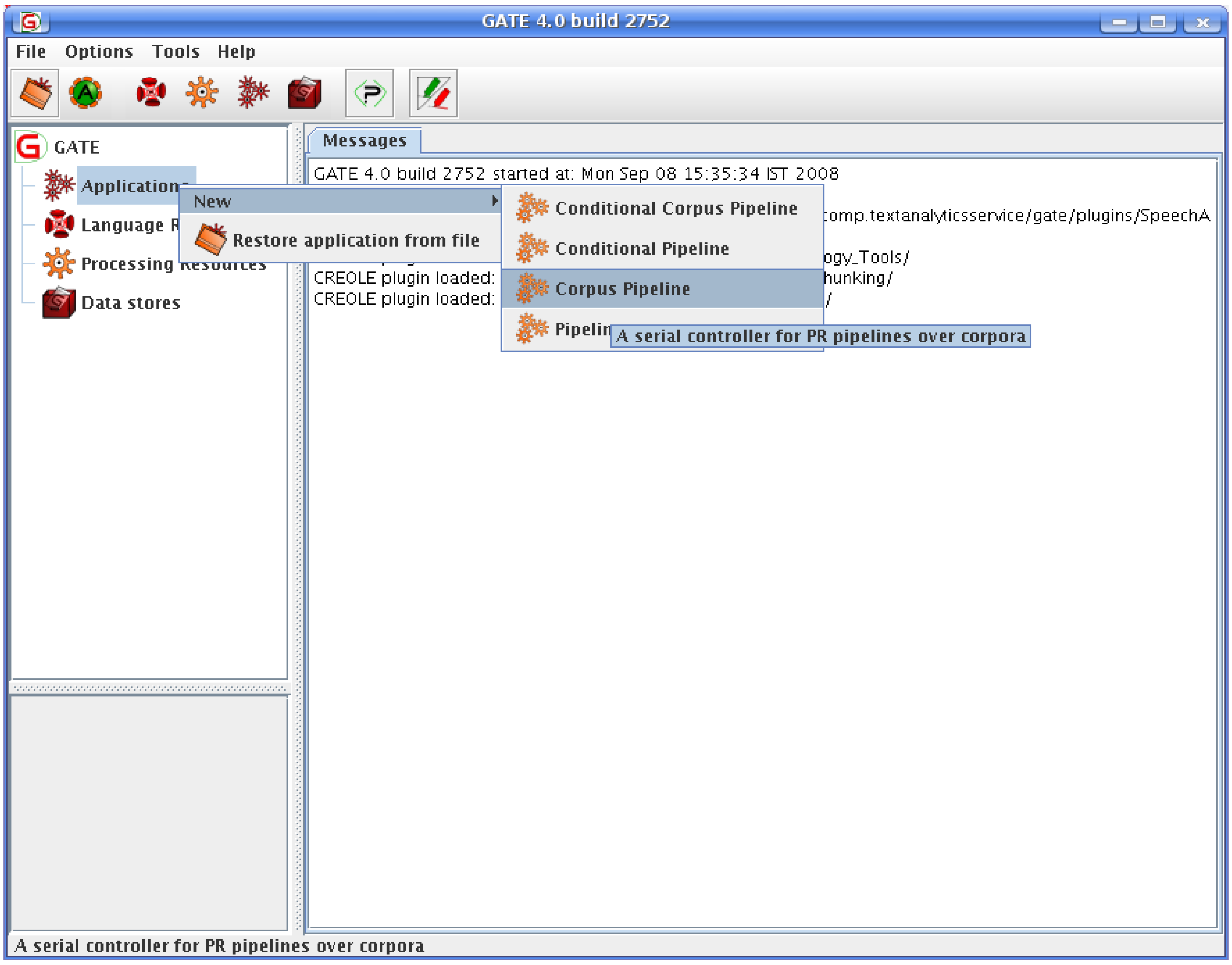
Take some time to explore the results.

Task 4: Domain specific IE

Now that we are off to a start, lets create our own GATE application which can extract simple relations from a business text.

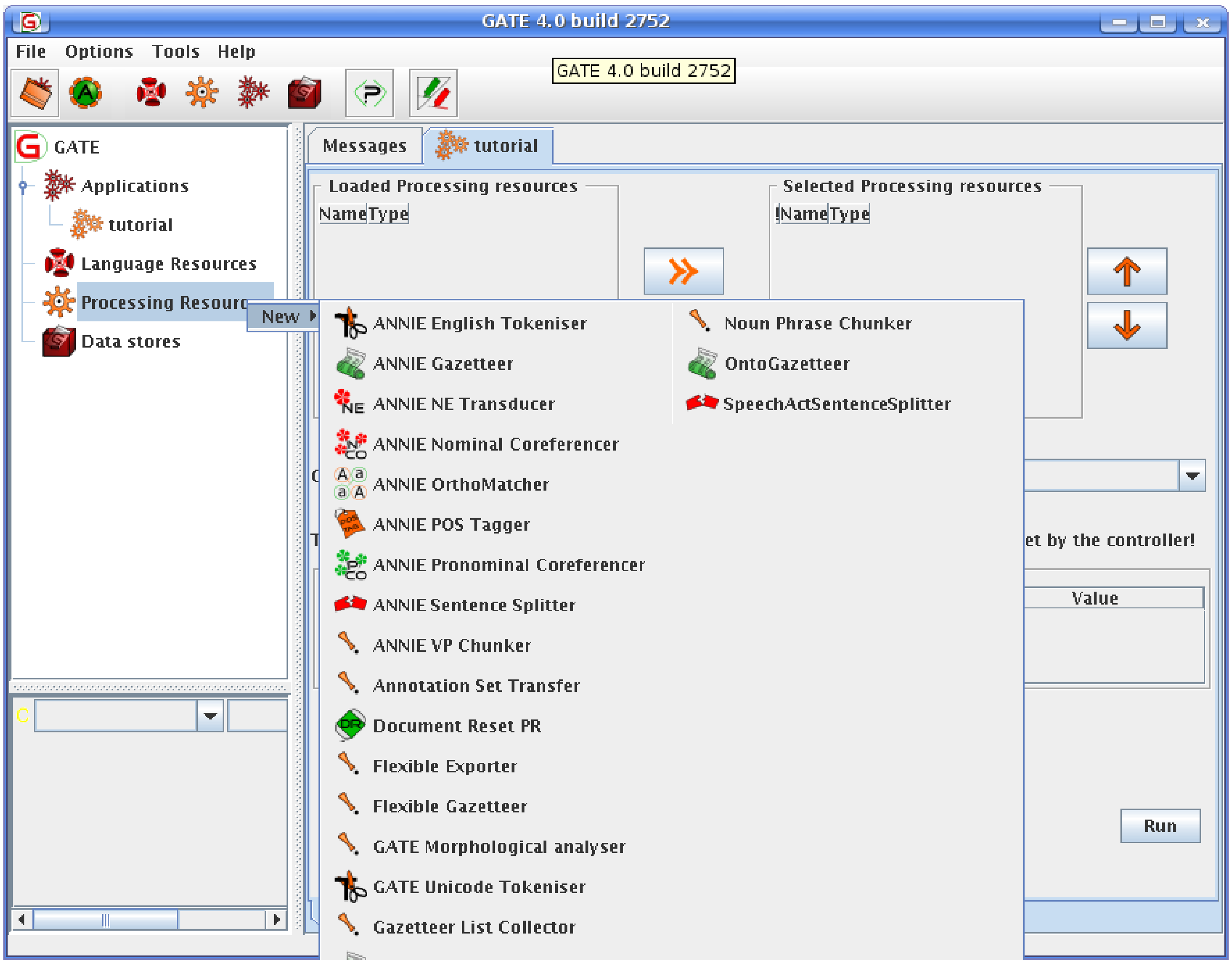
Step 1

Close your documents and the existing ANNIE application.
Create a Corpus pipeline in GATE.



```
GATE 4.0 build 2752 started at: Mon Sep 08 15:35:34 IST 2008
                                                    comp.textanalyticsservice/gate/plugins/SpeechA
                                                    ogy_Tools/
CREOLE plugin loaded:                               hunking/
CREOLE plugin loaded:                               /
```

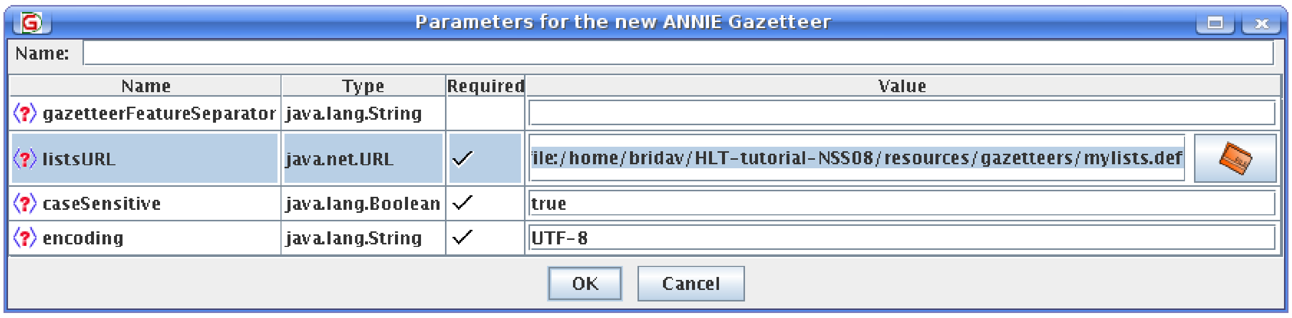
A serial controller for PR pipelines over corpora



Load the following Processing Resources (PRs) by right clicking on the Processing Resource Tree:

ANNIE English Tokeniser
ANNIE Gazetteer
ANNIE POS Tagger
ANNIE SentenceSplitter

Click Ok for all of the above PRs to accept the default values **EXCEPT the ANNIE Gazetteer.** See the ANNIE gazetteer settings below. Point the **listsURL attribute** to following path: file:<YOURHOMEDIR><MYTUTORIALDIR>/resources/gazetteers/mylists.def

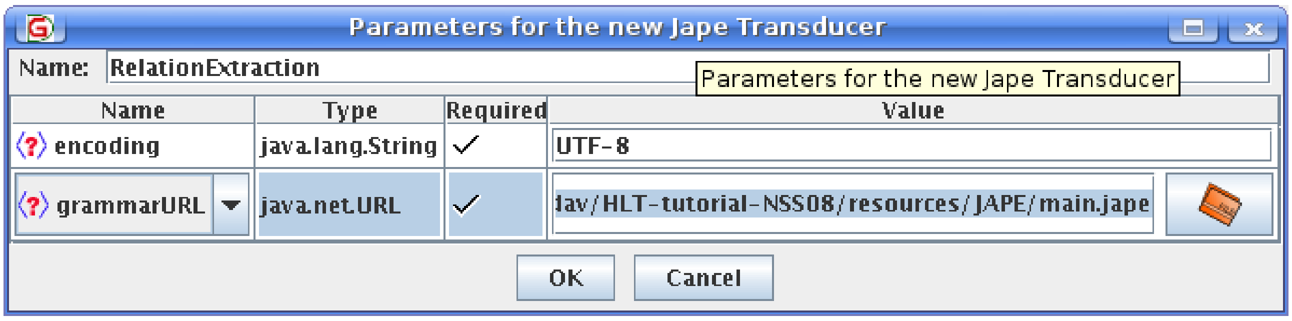

Parameters for the new ANNIE Gazetteer

Name:

Name	Type	Required	Value
gazetteerFeatureSeparator	java.lang.String		
listsURL	java.net.URL	✓	ile:/home/bridav/HLT-tutorial-NSS08/resources/gazetteers/mylists.def
caseSensitive	java.lang.Boolean	✓	true
encoding	java.lang.String	✓	UTF-8

OK Cancel

Now, create a new JAPE transducer PR.


Parameters for the new Jape Transducer

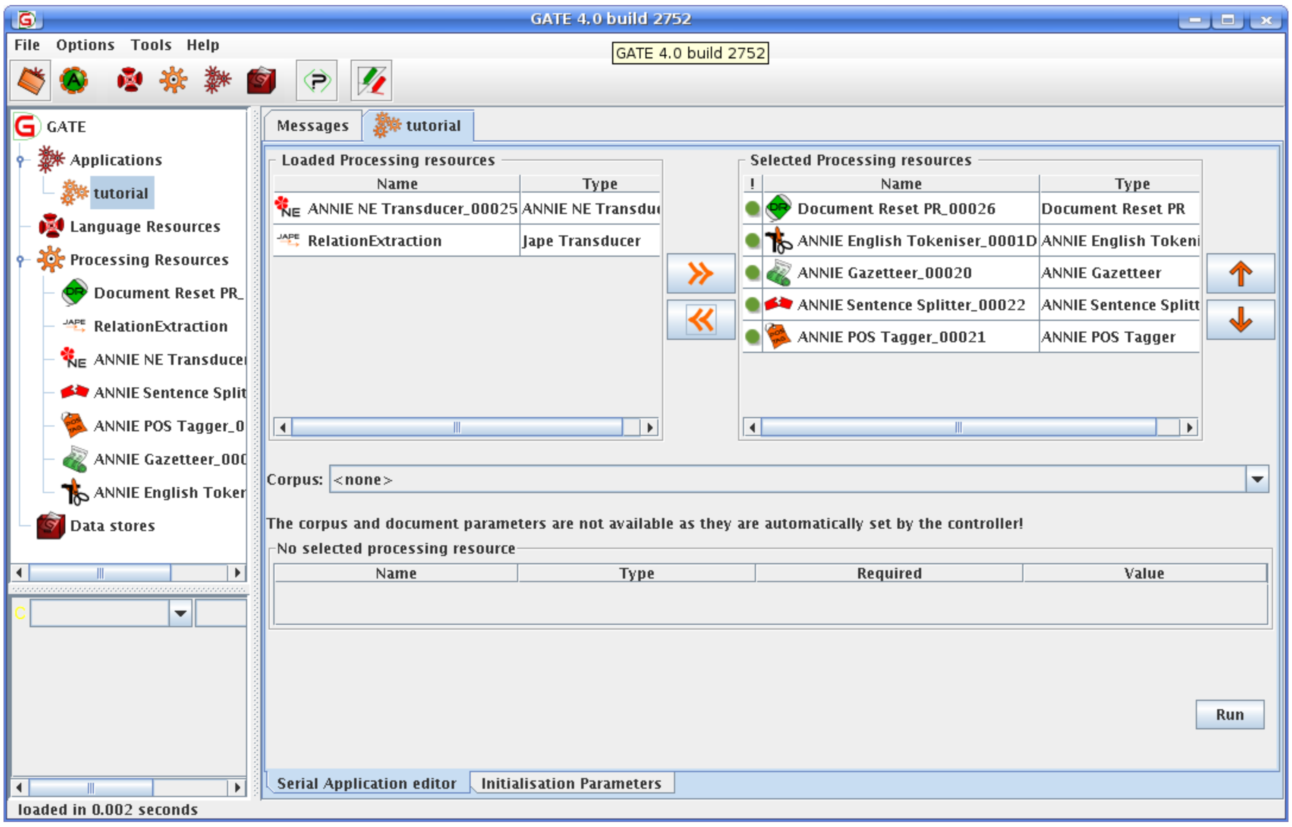
Name: RelationExtraction

Name	Type	Required	Value
encoding	java.lang.String	✓	UTF-8
grammarURL	java.net.URL	✓	dav/HLT-tutorial-NSS08/resources/JAPE/main.jape

OK Cancel

In the grammarURL attribute value, browse the material copied from within the Tutorial folder to this location:
file:/<YOURHOMEDIR><MYTUTORIALDIR>/resources/JAPE/main.jape

Using the green arrows in the Application Viewing pane for the “tutorial” pipeline. Move the PRs accross and “drop” them into the Corpus Pipeline.


GATE 4.0 build 2752

File Options Tools Help

GATE
- Applications
  - tutorial
- Language Resources
- Processing Resources
  - Document Reset PR_
  - RelationExtraction
  - ANNIE NE Transducer
  - ANNIE Sentence Split
  - ANNIE POS Tagger_0
  - ANNIE Gazetteer_000
  - ANNIE English Toker
- Data stores

Messages | tutorial

Loaded Processing resources

Name	Type
ANNIE NE Transducer_00025	ANNIE NE Transduc
RelationExtraction	Jape Transducer

Selected Processing resources

!	Name	Type
●	Document Reset PR_00026	Document Reset PR
●	ANNIE English Tokeniser_0001D	ANNIE English Tokeni
●	ANNIE Gazetteer_00020	ANNIE Gazetteer
●	ANNIE Sentence Splitter_00022	ANNIE Sentence Splitt
●	ANNIE POS Tagger_00021	ANNIE POS Tagger

Corpus: <none>

The corpus and document parameters are not available as they are automatically set by the controller!

No selected processing resource

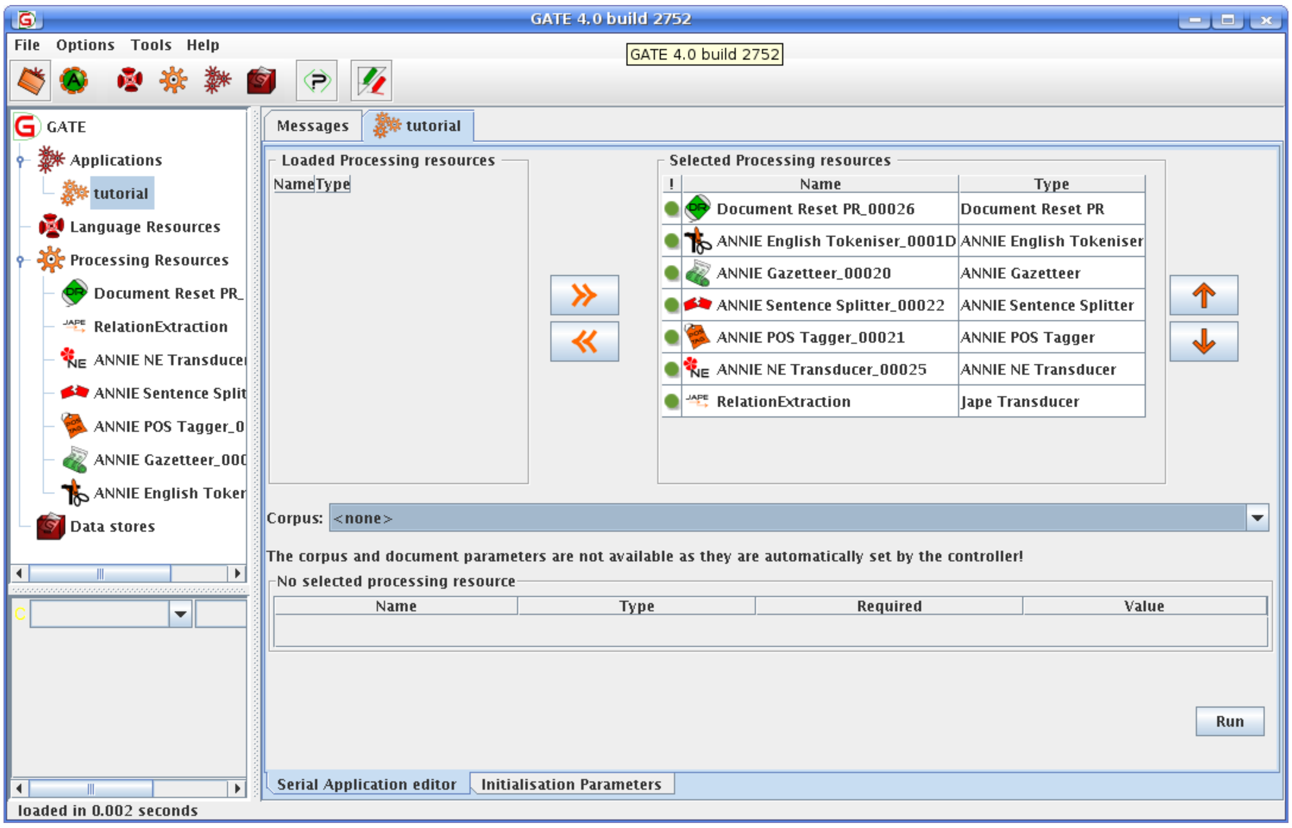
Name	Type	Required	Value

Run

Serial Application editor | Initialisation Parameters

loaded in 0.002 seconds

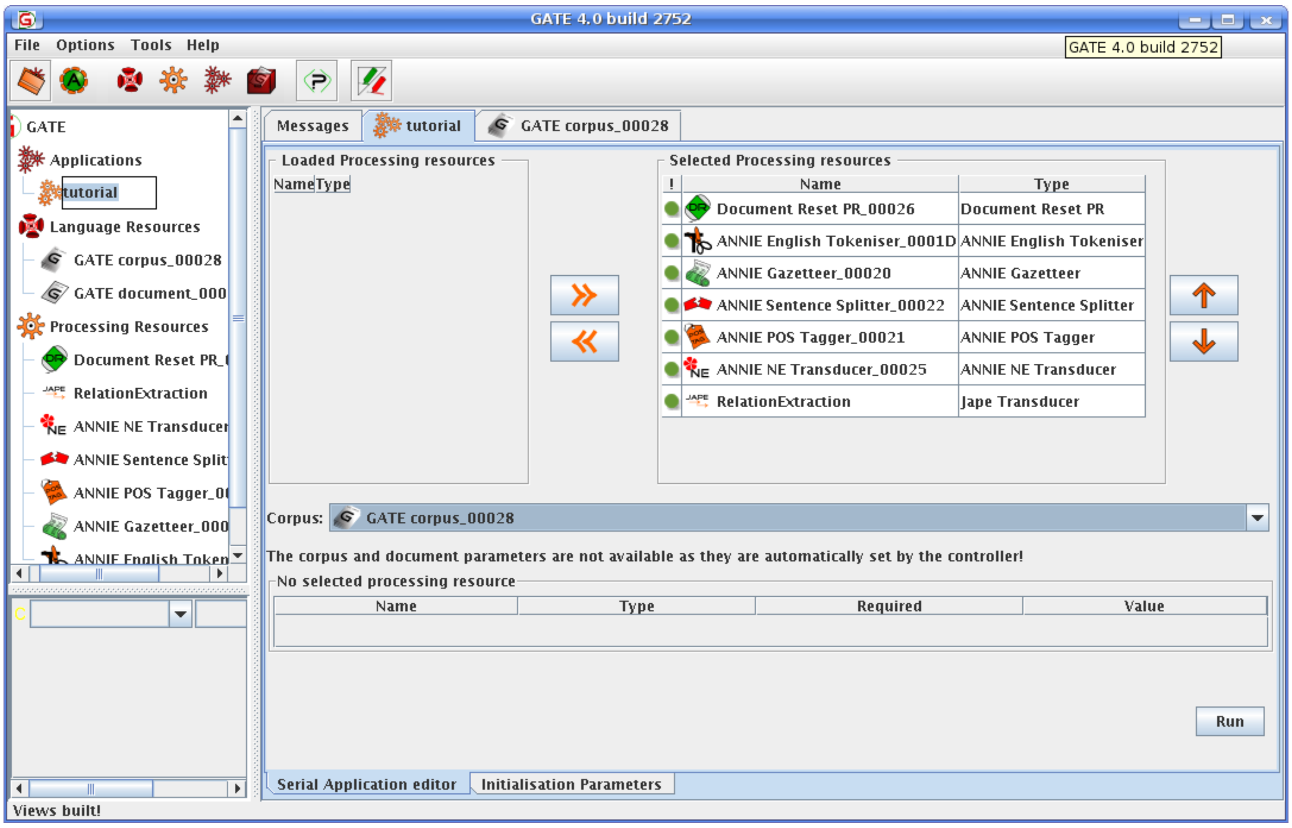
until your pipeline looks like the following below in the **correct order**:



!	Name	Type
●	Document Reset PR_00026	Document Reset PR
●	ANNIE English Tokeniser_0001D	ANNIE English Tokeniser
●	ANNIE Gazetteer_00020	ANNIE Gazetteer
●	ANNIE Sentence Splitter_00022	ANNIE Sentence Splitter
●	ANNIE POS Tagger_00021	ANNIE POS Tagger
●	ANNIE NE Transducer_00025	ANNIE NE Transducer
●	RelationExtraction	Jape Transducer

The corpus and document parameters are not available as they are automatically set by the controller!

Follow the same steps used in Task 3 to create a corpus.
In addition create a document, but this time use the following file from the tutorial folder:
file/<YOURHOMEDIR><MYTUTORIALDIR>/documents/BusinessText.txt
Add the GATE document to the GATE Corpus and set the Corpus in the "tutorial" pipeline. (Repeat of steps in Task 3)


GATE 4.0 build 2752

File Options Tools Help

GATE
- Applications
  - tutorial
- Language Resources
  - GATE corpus_00028
  - GATE document_000
- Processing Resources
  - Document Reset PR_(
  - RelationExtraction
  - ANNIE NE Transducer
  - ANNIE Sentence Split
  - ANNIE POS Tagger_0(
  - ANNIE Gazetteer_000
  - ANNIE English Token

Messages | tutorial | GATE corpus_00028

Loaded Processing resources

Name	Type

Selected Processing resources

!	Name	Type
●	Document Reset PR_00026	Document Reset PR
●	ANNIE English Tokeniser_0001D	ANNIE English Tokeniser
●	ANNIE Gazetteer_00020	ANNIE Gazetteer
●	ANNIE Sentence Splitter_00022	ANNIE Sentence Splitter
●	ANNIE POS Tagger_00021	ANNIE POS Tagger
●	ANNIE NE Transducer_00025	ANNIE NE Transducer
●	RelationExtraction	Jape Transducer

Corpus: GATE corpus_00028

The corpus and document parameters are not available as they are automatically set by the controller!

No selected processing resource

Name	Type	Required	Value

Run

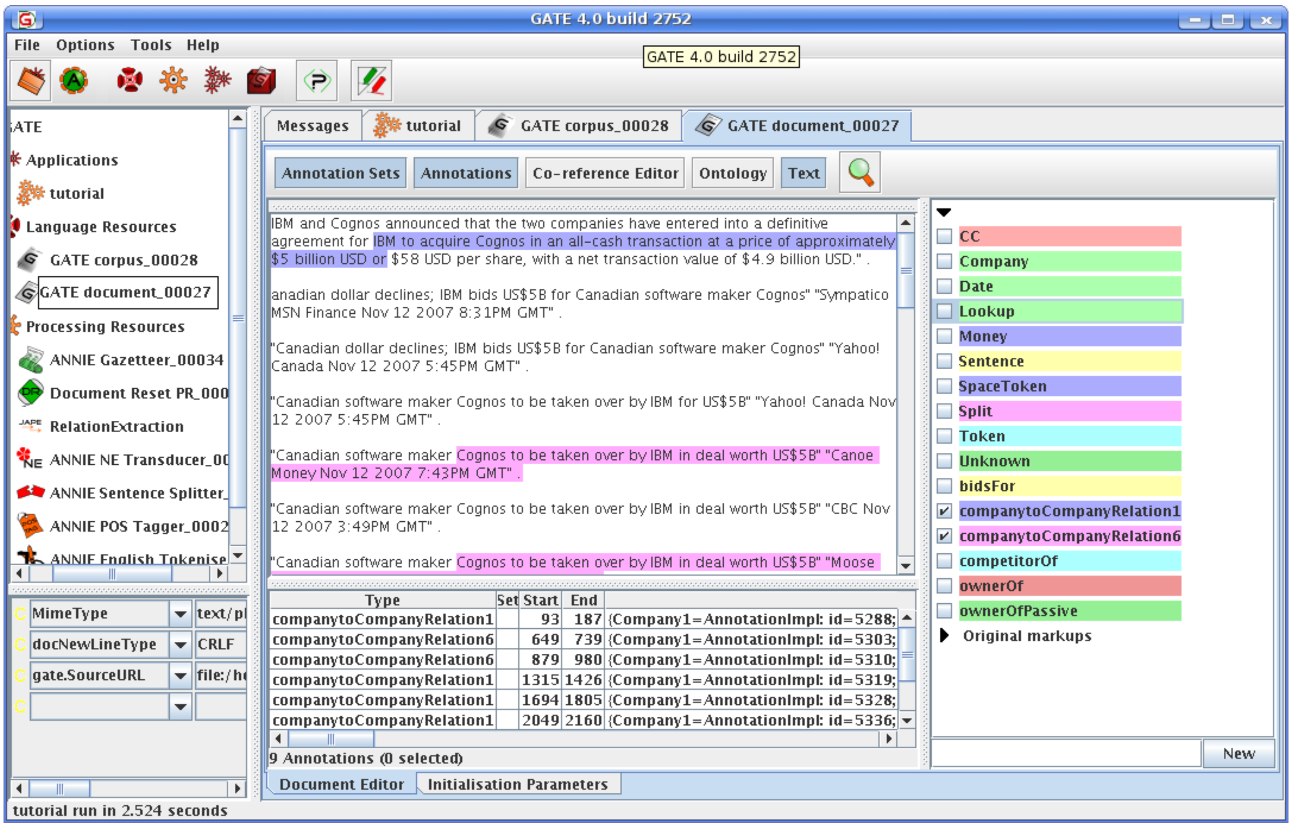
Serial Application editor | Initialisation Parameters

Views built!

Your Corpus pipeline should look like this.

Run the pipeline and wait until processing is completed. After this has occurred, double click on the GATE document (same steps as in Task 3).

Lets take a look at the annotations outputted:



MimeType	text/pl	
docNewLineType	CRLF	
gate.SourceURL	file:/ho	

IBM and Cognos announced that the two companies have entered into a definitive agreement for IBM to acquire Cognos in an all-cash transaction at a price of approximately $5 billion USD or $58 USD per share, with a net transaction value of $4.9 billion USD." .

anadian dollar declines; IBM bids US$5B for Canadian software maker Cognos" "Sympatico MSN Finance Nov 12 2007 8:31PM GMT" .

"Canadian dollar declines; IBM bids US$5B for Canadian software maker Cognos" "Yahoo! Canada Nov 12 2007 5:45PM GMT" .

"Canadian software maker Cognos to be taken over by IBM for US$5B" "Yahoo! Canada Nov 12 2007 5:45PM GMT" .

"Canadian software maker Cognos to be taken over by IBM in deal worth US$5B" "Canoe Money Nov 12 2007 7:43PM GMT" .

"Canadian software maker Cognos to be taken over by IBM in deal worth US$5B" "CBC Nov 12 2007 3:49PM GMT" .

"Canadian software maker Cognos to be taken over by IBM in deal worth US$5B" "Moose

Type	Set	Start	End	
companytoCompanyRelation1		93	187	{Company1=AnnotationImpl: id=5288;
companytoCompanyRelation6		649	739	{Company1=AnnotationImpl: id=5303;
companytoCompanyRelation6		879	980	{Company1=AnnotationImpl: id=5310;
companytoCompanyRelation1		1315	1426	{Company1=AnnotationImpl: id=5319;
companytoCompanyRelation1		1694	1805	{Company1=AnnotationImpl: id=5328;
companytoCompanyRelation1		2049	2160	{Company1=AnnotationImpl: id=5336;

9 Annotations (0 selected)

Notice there are two annotation sets "clicked on" in the GUI – **companytoCompanyRelation1** and **companytoCompanyRelation6**

Both relations are concerned with relations of type acquisition between **Company** annotations. **companytoCompany1** is concerned with **Company1 acquiring Company2** and **companytoCompany6** concerns acquiring relations where the trigger phrases are in the passive tense i.e. "IBM was taken over by Cognos".

Lets have a little more look in detail at the JAPE grammars which do all the work.
Open the file:<YOURHOMEDIR><MYTUTORIALDIR>/resources/JAPE/ main.jape

```
MultiPhase:    RelationExtractionGrammars
Phases:
preprocessing1
preprocessing2
acquireRelation
```

Now Lets take a closer look at file

<YOURHOMEDIR><MYTUTORIALDIR>/resources/JAPE/preprocessing2.jape

```
/*
*  preprocessing2.jape
*  Copyright (c) 2007- , DERI, National University of Ireland,Galway.
*  Brian Davis, 12 December 2007
*  Project: Training - Prototype
*  $Id: preprocessing2.jape,v 1.0 2007/12/12 19:24:00 GMT
*
* The Phase seperates Annotations of Company relations from Lookup Annotations
*/

Phase: preprocessing2
Input: Lookup

Rule: Preprocess1
(
{Lookup.majorType==acquires_verb}
):acquiresVerb

-->
:acquiresVerb
  {
     gate.AnnotationSet matchedAnns=
    (gate.AnnotationSet)bindings.get("acquiresVerb");
    gate.FeatureMap newFeatures= Factory.newFeatureMap();
    newFeatures.put("rule","Preprocess1");

annotations.add(matchedAnns.firstNode(),matchedAnns.lastNode(),"ownerOf", newFeatures);
  }


/* - acquires, took over */
```

As mentioned earlier JAPE consists of collection of rules each one consisting of a set of LHS --> RHS rules or <Pattern> --> {action } rules. The LHS of a JAPE rule uses Regular expressions (|, ?, *, +) over annotations while the RHS consists of a block of JAVA code for annotation manipulation. In the above rule
`acquiresVerb` is a binding varaiable for the Lookup annotation (Gazetteer Lookup). The binding variable is cast into an `AnnotationSet` object (`matchedAnns`). A `FeatureMap` object is created to store additional linguistic information, other annotations and/or in case this debugging information (i.e. If I want to know the name of the Jape rule that fired to create this annotation.) FeatureMaps store abitary attribute/value pairs as type `String/Object Java Objects` respectively. Hence the value of a Feature can be anything. Finally, in the last line we create and add a new annotation to the default annotationSet `annotations`.

`annotations.add(matchedAnns.firstNode(),matchedAnns.lastNode(),"ownerOf", newFeatures);`

The four parameters of the `add method are:`

The (1) start and the end (2) positions of the `matchedAnns` annotation,
(3) the name of the new annotation as a `String literal, in this case "ownerOf"` and (4) the `FeatureMap` object, `newFeatures.`

Now a new annotation `"OwnerOf"` has been created, which we will manipulate at a later stage.

Now Lets take a closer look at file <YOURHOMEDIR><MYTUTORIALDIR>/resources/JAPE/acquireRelation.jape

```
/*
* acquireRelation.jape
*
* Copyright (c) 2007- , DERI, National University of Ireland, Galway.
*  Brian Davis, 12 December 2007
*  Project: Lion - Prototype
*  $Id: acquireRelation.jape,v 1.0 2007/12/12 16:47:00 GMT
*/

Phase: CompanyRelations

Input: Company ownerOf Split CC



Rule: CompanytoCompanyRelation1

(

({Company}):c1 ({ownerOf}):v ({Company}):c2 ({Split}|{CC})

):companytoCompanyRelation1



-->

:companytoCompanyRelation1

  {

    gate.AnnotationSet matchedCompanies=(gate.AnnotationSet) bindings.get("c1");
    Annotation company1=matchedCompanies.iterator().next();
    gate.AnnotationSet matchedCompanies2=(gate.AnnotationSet) bindings.get("c2");
    Annotation company2=matchedCompanies2.iterator().next();
    gate.AnnotationSet matchedVerb=(gate.AnnotationSet) bindings.get("v");
    Annotation verb=matchedVerb.iterator().next();
    gate.AnnotationSet matchedAnns= (gate.AnnotationSet)
    bindings.get("companytoCompanyRelation1");
    gate.FeatureMap newFeatures= Factory.newFeatureMap();
    newFeatures.put("Company1",company1);
    newFeatures.put("ownerOf",verb);
    newFeatures.put("Company2",company2);
    newFeatures.put("rule","CompanytoCompanyRelation1");
    annotations.add(matchedAnns.firstNode(),matchedAnns.lastNode(),"companytoCompanyRelat
ion1", newFeatures);



}
```

This Jape file is responsible for extracting relations of type `companytoCompanyRelation1` and `companytoCompanyRelation6` – relations of type "ownership" between two `Company` annotations. The rules are similar to the above file with some notable exceptions:

In a Jape file you can specify the input annotations to a transducer much like you would pass parameters to a procedure or a function in any other programming language.

Observe The line:

Input: Company ownerOf Split CC

states that the JAPE transducer should only consider the annotations of type
Company #Companies already identified by gazetteer lookup and split into
    Company Annotations i.e "IBM", "Microsoft"

ownerOf #trigger phrases of type "ownerOf" already identified by gazetteer
    lookup and split into seperate "ownerOf" Annotations

Split #annotations created by the Sentence Splitter - ?,.,\t,\r

CC #cetain types of Coordinating conjunctions between clauses – or, and, ","

Note that due to specifying the Input, all other annotations will be ignored i.e. Lookup, Token etc. This also justifies seperating Lookup annoations into Company etc – as JAPE patterns for relation extraction based on Lookup annotations only, would overgeneralise and the rules would never fire.

Now lets look at the LHS JAPE pattern

```
(

({Company}):c1 ({ownerOf}):v ({Company}):c2 ({Split}|{CC})

):companytoCompanyRelation1
```

#Annotations are surrounded by {...}
#Patterns or Annotations to be bound are to labels using (...):bindVar

With

```
({Split}|{CC})
```

You have a simple RE pattern which implies match either a Split or a CC annotation. The entire pattern is bound by the following label/binding variable companytoCompanyRelation1 in the following manner:

```
 (<PATTERN>):bindVar
-->
 :bindVar { JAVA BLOCK }
```

Consequently this rule will match:

"IBM to acquire Cognos in an all-cash transaction at a price of approximately $5 billion USD or"

Now we will move The RHS action rule of this JAPE example:

Some explanations:

```
gate.AnnotationSet matchedCompanies=(gate.AnnotationSet) bindings.get("c1");
      Annotation company1=matchedCompanies.iterator().next();
```

#This piece of code converts the binding variable for the `Company` annotation into an `AnnotationSet` called `matchedCompanies`. Since I am only interested in one `Company`, I used the `AnnotationSet's iterator method` to give me the first `Annotation` within the `AnnoationSet`. This process is repeated from the `v` and `c2 labels` bound to the JAPE LHS.

```
gate.AnnotationSet matchedAnns= (gate.AnnotationSet)
    bindings.get("companytoCompanyRelation1");
```

#We apply the same process again, but this time we create a new annotation set that spans the entire relation containing the `Company ownerOf Company` annotations.

```
    gate.FeatureMap newFeatures= Factory.newFeatureMap();
    newFeatures.put("Company1",company1);
    newFeatures.put("ownerOf",verb);
    newFeatures.put("Company2",company2);
    newFeatures.put("rule","CompanytoCompanyRelation1");
```

Now we create a new `FeatureMap` and attribute\value pairs. Recall that value of an attribute can be any Java Object. In the above example, the new annotations for `ownerOf` and the companies involved in the relation are "chained" or embedded within the `companytoCompanyRelation1` annotation using FeatureMaps.

```
  annotations.add(matchedAnns.firstNode(),matchedAnns.lastNode(),"companytoCompanyRelatio
n1", newFeatures);
```

Finally just as before we add our `new companytoCompanyRelation`1 to the default annotationSet `annotations`. Later we will learn how to access these annoations programatically in GATE and map them into RDF triples.

BUT First lets play a little with JAPE -

Task 5: X marks the spot !

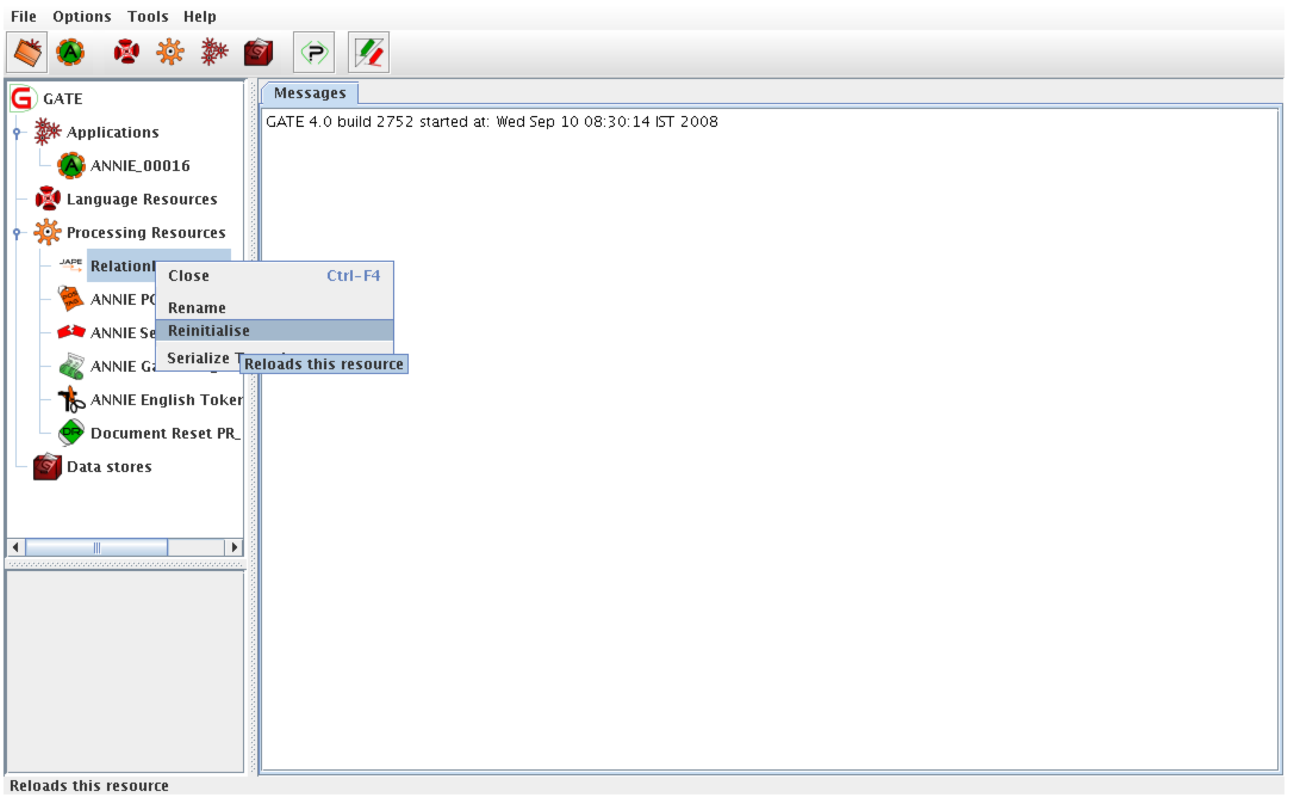
Create a new JAPE rule to extract `isCustomerOf` relations between companies.

Use the following file:**<YOURHOMEDIR><MYTUTORIALDIR> resources/JAPE/myjape.jape**

```
/*

* myjape.jape

*

*  Copyright (c) 2008- , Some University .

*  Somebody, 09 September 2008

*  Project XXXXXXX

*/


Phase: CompanyRelations2

Input: Company customerOf Split

Options: control=all


Rule: CompanytoCompanyRelation2

(

({Company}):c1({customerOf}):v({Company}):c2 {Split}

):companytoCompanyRelation2


-->

:XXXXXXXXXXX

  {



      gate.AnnotationSet matchedCompanies=(gate.AnnotationSet)
bindings.get("c1");

    Annotation company1=matchedCompanies.iterator().next();

    gate.AnnotationSet matchedCompanies2=(gate.AnnotationSet)
bindings.get("c2");

    Annotation company2=matchedCompanies2.iterator().next();

    gate.AnnotationSet matchedVerb=(gate.AnnotationSet) bindings.get("v");

    Annotation verb=matchedVerb.iterator().next();

    gate.AnnotationSet matchedAnns= (gate.AnnotationSet)
bindings.get("XXXXXXXXXX");

    gate.FeatureMap newFeatures= Factory.newFeatureMap();
    newFeatures.put("Company1",company1);

    newFeatures.put("customerOf",verb);

    newFeatures.put("Company2",company2);

      newFeatures.put("rule","CompanytoCompanyRelation2");

      annotations.add(XXXXXXXXX);

}


/* Company 2 Company relation -  buys, purchases, is a customer of */
```

Once you have finished – reinitialize your JAPE transducer – but dont forget to amend your main.jape file to include your new Grammar.

Reinitialize your JAPE transducers like so:


GATE 4.0 build 2752 started at: Wed Sep 10 08:30:14 IST 2008

More Advanced Problems:

What happens if we have false positives i.e.

Oracle's Larry Ellison has long threatened to acquire BEA Systems, finally making an offer on October 12, 2007, however this was rejected by BEAs

Board of Directors.

SAP no longer plans to buy Business Objects.

Write a JAPE file to handle the following exceptions: contradiction and negation.

Tips

You will need to:
add new gazeteer entries
Modify preprocessing2.jape
Modify acquireRelation.jape

Hint one way to do it is to create a *Contrst* annotation (for contrastive conjunction). A contrastive conjunction implies the preceeding clause will be contradicted by the proceeding subclause

We could create a new rule as such:

```
Phase: CompanyRelations2

Input: Company customerOf Split Contrst

Options: control=appelt


Priority:25

Rule: CompanytoCompanyRelationNeg

(

({Company}):c1 ({ownerOf}):v ({Company}):c2 ({Contrst})

):companytoCompanyRelationNeg

--> {}
```

and add it into

<YOURHOMEDIR><MYTUTORIALDIR>/resources/JAPE/acquireRelation.jape

Make sure it preceeds the old rule. Why?

Notice in the file we currently have set the controll of this jape grammar to *all*. This means all rules will fire! We can change it to:

```
Options: control=appelt
```

This means the rule with the longest match will fire first! In addtion you can specify the the prioity of rules to fired (just to be on the save side) i.e.

```
Priority:25
```

You do not need to change the priority of the other rules in this case since unless specified, the default priority for all rules is -1 which is the lowest priority. Hence priorities are positive integers.

Finally you'll notice --> {}. This is an empty Java block, hence when it is fired it will do nothing. More importanly JAPE will consume the annotations and continue matching from the next offset. If the rule does not match then the next set of **possitive** rules can proceed in priority to match.

You could use the negative operators (GATE 5) in JAPE to colapse the rule `CompanytoCompanyRelationNeg` into `CompanytoCompanyRelation2`

```
Rule: CompanytoCompanyRelation2

(

({Company}):c1({customerOf}):v({Company}):c2 !({Contrst}){Split}
)
```

Be warned though that negating `Contrst` tells JAPE to match any other annotation except `Contrst`, hence in this case you could match `Company , customerOf or Split` which could lead to overgeneration of your rule or unepected results. If you are unsure which approach to use, pick the first one (<PATTERN> --> {}).

For Testing use:
file/<YOURHOMEDIR><MYTUTORIALDIR>/documents/BusinessTextDecoy.txt

Finally, those of you with a theoretical computational linguistic's background will not doubt express concerns over the oversimplicity of the rules to handle negatio/contradiction above. In particular, it would difficult to account for all syntactic variations of negation before or after the verb trigger phrase `customerOf`. Furthermore JAPE patterns have the equivalent recognition power of a regular language and one might presume this would insuffficent for your needs. One would assume   JAPE cannot backtrack, but in fact some degree of backtracking is permitted since the annotation sets you have created are always accessable via the right hand side Java block, hence you can always access the memory of your parse/annotation information and delete, amend, update annotation sets or specific annotations. Futhermore you can check for intersections with your matched annotation span and other annotations which is useful if you cannot account for the syntactic ordering of some linguistic phenomenon (HINT for negation:-)) . Futhermore, there is no reason not to use other available parsers within GATE such as, NP and VP chunkers, Minipar parser, Standford parser, SUPPLE parser, RASP parser, depending on your NLP

task.
One useful advantage is that the annoations created by such parsers as Minipar, NP and VP chunkers etc can be later manipulated in JAPE.

A the JAVA code is available to build in the <YOURHOMEDIR><MYTUTORIALDIR>/sources directory please inspect the README file for instructions to compile and run the program.

Thank You for your time and enjoy playing with GATE!

References

[1]GATE User Guide See http://gate.ac.uk/sale/tao/split.html
[2]Software Architecture for Language Engineering
http://gate.ac.uk/sale/thesis/index.html
[3]http://www.alphaworks.ibm.com/tech/lrw
[4]http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.36.7858