Module 10:
Advanced GATE Applications

© The University of Sheffield, 1995-2013
This work is licenced under the Creative Commons Attribution-NonCommercial-ShareAlike Licence

About this tutorial

- This tutorial will be a mixture of explanation, demos and hands-on work
- Things for you to try yourself are in red
- Example JAPE code is in `blue`
- It assumes basic familiarity with the GATE GUI and with ANNIE and JAPE; you don't need Java expertise
- Your hands-on materials are in module-10-advanced-ie/hands-on/
- There you'll find a **corpus** directory containing documents, and a **grammar** directory containing JAPE grammar files, and various other files.
- Completing the hands-on tasks will help you in the exam....

Finding available resources

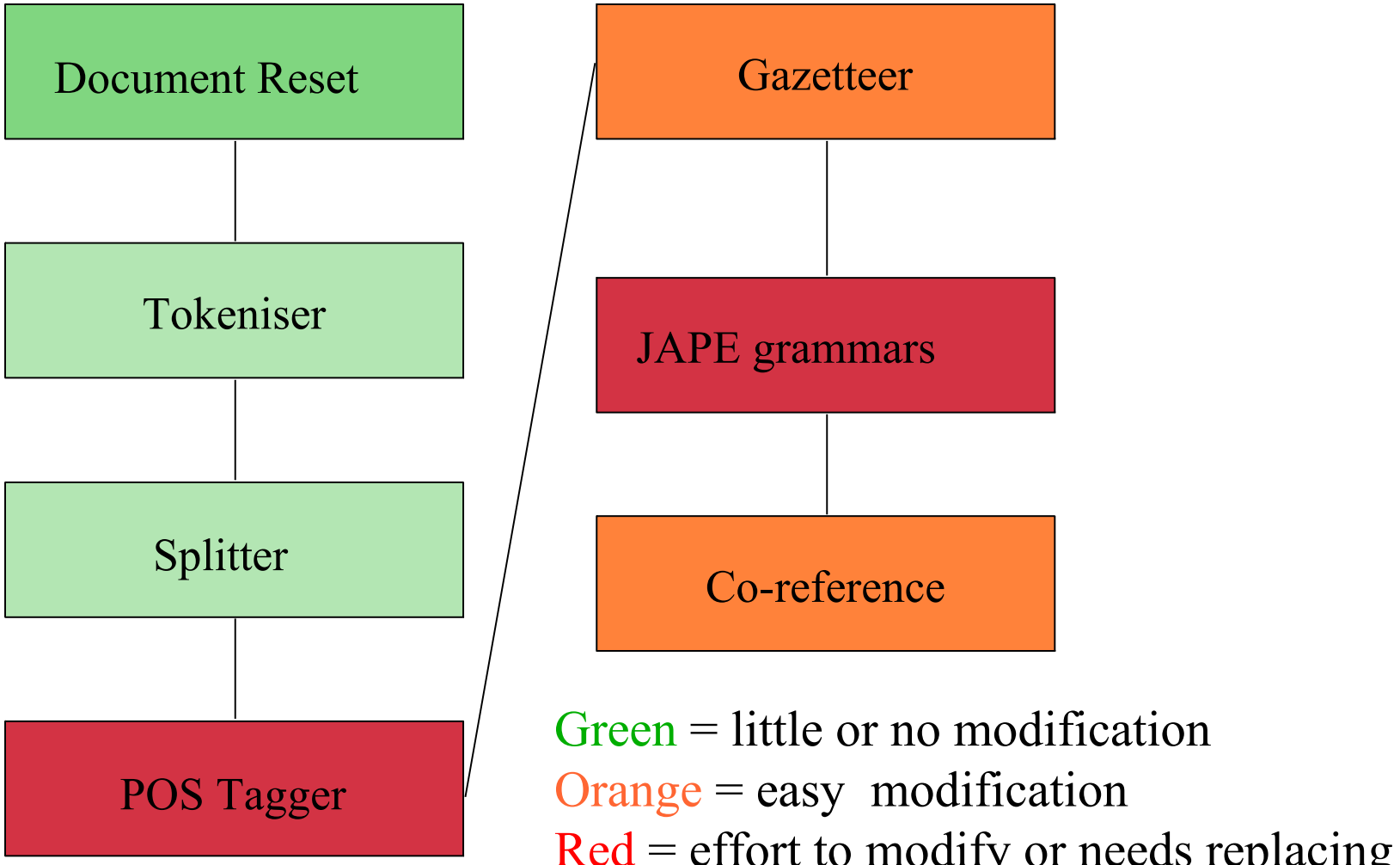
- When creating an IE system for new languages, it's easiest to start with ANNIE and then work out what needs adapting
- Check the resources in GATE for your language (if any)
  - Check the gate/plugins directory (hint: the language plugins begin with Lang_*)
  - Check the user guide for things like POS taggers and stemmers which have various language options
- Check which PRs you can reuse directly from ANNIE
  - Existing tokeniser and sentence splitter will work for most European languages. Asian languages may require special components.
- Collect any other resources for your language, e.g POS taggers. These can be implemented as GATE plugins.

Which resources need modifying?

We can divide the PRs into 3 types depending on how much modification they need to work with other languages:

- **language-independent**: work with different languages with little or no modification
- **easily modifiable**: can be easily modified for a different language with little programming skill
- **language-dependent**: these need to be replaced by an entirely new PR

How easy is ANNIE to modify?



Language-independent resources

- ANNIE PRs which are totally language-independent are the **Document Reset** and **Annotation Set Transfer**
- They can be seen as “language-agnostic” as they just make use of existing annotations with no reference to the document itself or the language used
- The **tokeniser** and **sentence splitter** are (more or less) language-independent and can be re-used for languages that have the same notions of token and sentence as English (white space, full stops etc)
- Make sure you use the Unicode tokeniser, not the English tokeniser (which is customised with some English abbreviations etc)
- Some tweaking could be necessary - these PRs are easy to modify (with no Java skills needed)

Easily modifiable resources

- **Gazetteers** are normally language-dependent, but can easily be translated or equivalent lists found or generated
  - Lists of numbers, days of the week etc. can be translated
  - Lists of cities can be found on the web
- Gazetteer modification requires no programming or linguistic skills
- The **Orthomatcher** will work for other languages where similar rules apply, e.g. John Smith --> Mr Smith
- Might need modification in some cases: some basic Java skills and linguistic knowledge are required

Language-dependent resources

- **POS taggers** and **grammars** are highly language-dependent
- If no POS tagger exists, a hack can be done by replacing the English lexicon for the Hepple tagger with a language-specific one
- Some grammar rules can be left intact, but many will need to be rewritten
- Many rules may just need small modifications, e.g. component order needs to be reversed in a rule
- Knowledge of some linguistic principles of the target language is needed, e.g. agglutination, word order etc.
- No real programming skills are required, but knowledge of JAPE and basic Java are necessary

Adding a POS tagger for a new language

- If you already have a POS tagger for your language with a Java API, you can write a "wrapper" PR for it
  - This enables you to feed sentences/tokens to the tagger, and map the output back to GATE annotations
  - See the Parser_Stanford plugin for an example of this.
- If you have a POS-tagged corpus, you could translate it into "traditional" tagged format with one line per sentence, e.g.

  The_DT cat_NN sat_VBD on_IN the_DT mat_NN ._.

  - You can then use the resulting trained model as a parameter for the LingPipe POS Tagger PR
  - This is how we POS-tagged Bulgarian in GATE

Tree Tagger

- Language-independent POS tagger supporting English, French, German, Spanish in GATE
- Needs to be installed separately
- Also supports Italian and Bulgarian, but not in GATE
- Tagger framework should be used to run the TreeTagger
- This provides a generic wrapper for various taggers
- In addition to TreeTagger, sample applications for
  - GENIA (English biomedical tagger)
  - HunPos (English and Hungarian)
  - Stanford Tagger (English, German and Arabic)
- More details in the GATE User Guide

Extra hands-on: TreeTagger

- Follow the instructions in the GATE User Guide to download and install the TreeTagger
- Try it out on some sample text for the relevant language (you can use Google to find documents in different languages)
- Pay very close attention to ALL the steps mentioned in the instructions

Named Entity Recognition without Training Data on a Language you don't speak:

The Surprise Language Exercise

An IE system for Cebuano

- On 4 March 2003, a bomb exploded in Davao City. The President of the Philippines classified this event as a terrorist attack.
- 24 hours later, Cebuano was announced as the language to be used in an experiment to create tools and resources for a surprise language.
- Within 4 days, we had developed a POS tagger for Cebuano, and within 7 days, we developed an NE system for Cebuano with 77.5% F measure.
- We did this, having never heard of the language, with **no native speaker** and **no training data**.
- We also used essentially the manpower of only 1 person

Are we mad?

- Quite possibly
- At least, most people thought we were mad to attempt this, and they’re probably right...
- Our results, however, are genuine.
- It's a good example of rough and ready adaptation of our basic IE resources to a new language
- So, what is it all about, and how on earth did we do it?

The Surprise Language Exercise

- In the event of a national emergency, how quickly could the NLP community build tools for language processing to support the US government?
- Typical tools needed: IE, MT, summarisation, CLIR
- Main experiment in June 2003 gave sites a month to build such tools
- Dry run in March 2003 to explore feasibility of the exercise.

Dry Run

Ran from 5-14 March as a test to:

- see how feasible such tasks would be
- see how quickly the community could collect language resources
- test working practices for communication and collaboration between sites

What on earth is Cebuano?

- Spoken by 24% of the Philippine population and the lingua franca of the S. Philippines (incl. Davao City)
- Classified by the LDC as a language of “medium difficulty”.
- Very few resources available (large scale dictionaries, parallel corpora, morphological analyser etc)
- But Latin script, standard orthography, words separated by white space, many Spanish influences and a lot of English proper nouns make it easier….

Named Entity Recognition

- For the dry run, we worked on resource collection and development for NE.
- Useful for many other tasks such as MT, so speed was very important.
- Test our claims about ANNIE being easy to adapt to new languages and tasks.
- Rule-based meant we didn't need training data.
- But could we write rules without knowing any Cebuano?

Resources

- Collaborative effort between all participants, not just those doing IE
- Collection of general tools, monolingual texts, bilingual texts, lexical resources, and other info
- Resources mainly from web, but others scanned in from hard copy

Text Resources

- Monolingual Cebuano texts were mainly news articles (some archives, others downloaded daily)
- Bilingual texts were available, such as the Bible, but not very useful for NE recognition because of the domain.
- One news site had a mixture of English and Cebuano texts, which were useful for mining.

Lexical Resources

- Small list of surnames
- Some small bilingual dictionaries (some with POS info)
- List of Philippine cities (provided by Ontotext)
- But many of these were not available for several days

Other Resources

- Infeasible to expect to find Cebuano speakers with NLP skills and train them within a week
- But extensive email and Internet search revealed several native speakers willing to help
- One local native speaker found - used for evaluation
- yahoogroups Cebuano discussion list found, leading to provision of new resources etc.

Adapting ANNIE for Cebuano

- Default IE system is for English, but some modules can be used directly
- Used tokeniser, splitter, POS tagger, gazetteer, NE grammar, orthomatcher (coreference)
- Splitter and orthomatcher unmodified
- Added tokenisation post-processing, new lexicon for POS tagger and new gazetteers
- Modified POS tagger implementation and NE grammars

Tokenisation

- Used default Unicode tokeniser
- Multi-word lexical items meant POS tags couldn't be attached correctly
- Added post-processing module to retokenise these as single Tokens
- Created gazetteer list of such words and a JAPE grammar to combine Token annotations
- Modifications took approx. 1 person hour

POS tagger

- Used Hepple tagger but substituted Cebuano lexicon for English one
- Used empty ruleset since no training data available
- Used default heuristics (e.g. return NNP for capitalised words)
- Very experimental, but reasonable results

Evaluation of Tagger

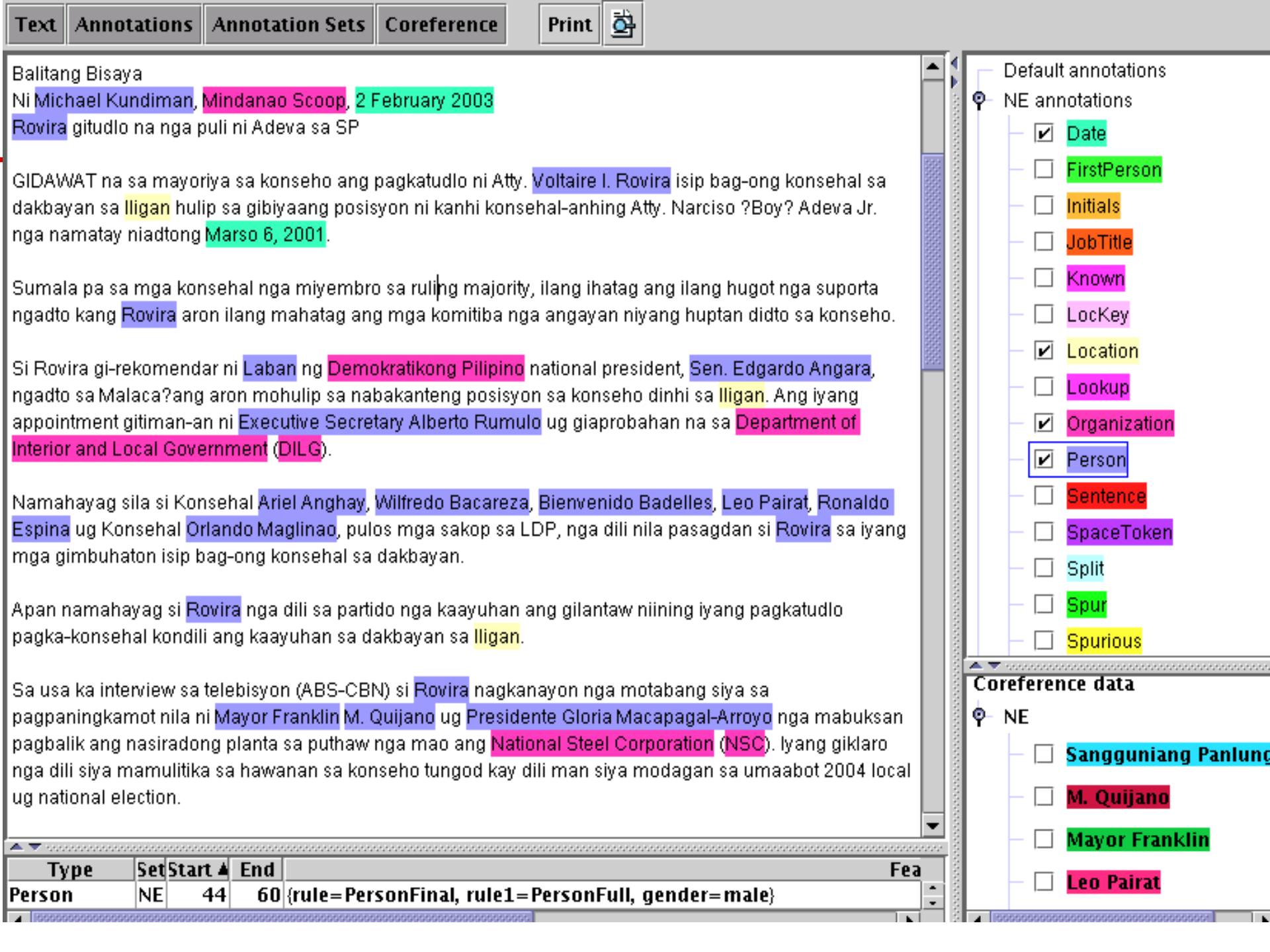
- No formal evaluation was possible
- Estimate around 75% accuracy
- Created in 2 person days
- Results and a tagging service made available to other participants

Gazetteer

- Perhaps surprisingly, very little info on Web
- Mined English texts about Philippines for names of cities, first names, organisations ...
- Used bilingual dictionaries to create “finite” lists such as days of week, months of year..
- Mined Cebuano texts for “clue words” by combination of bootstrapping, guessing and bilingual dictionaries
- Kept English gazetteer because many English proper nouns and little ambiguity

NE grammars

- Most English JAPE rules based on POS tags and gazetteer lookup
- Grammars can be reused for languages with similar word order, orthography etc.
- No time to make detailed study of Cebuano, but very similar in structure to English
- Most of the rules left as for English, but some adjustments to handle especially dates


Text
Annotations
Annotation Sets
Coreference
Print
Balitang Bisaya
Ni Michael Kundiman, Mindanao Scoop, 2 February 2003
Rovira gitudlo na nga puli ni Adeva sa SP
GIDAWAT na sa mayoriya sa konseho ang pagkatudlo ni Atty. Voltaire I. Rovira isip bag-ong konsehal sa dakbayan sa Iligan hulip sa gibiyaang posisyon ni kanhi konsehal-anhing Atty. Narciso ?Boy? Adeva Jr. nga namatay niadtong Marso 6, 2001.
Sumala pa sa mga konsehal nga miyembro sa ruling majority, ilang ihatag ang ilang hugot nga suporta ngadto kang Rovira aron ilang mahatag ang mga komitiba nga angayan niyang huptan didto sa konseho.
Si Rovira gi-rekomendar ni Laban ng Demokratikong Pilipino national president, Sen. Edgardo Angara, ngadto sa Malaca?ang aron mohulip sa nabakanteng posisyon sa konseho dinhi sa Iligan. Ang iyang appointment gitiman-an ni Executive Secretary Alberto Rumulo ug giaprobahan na sa Department of Interior and Local Government (DILG).
Namahayag sila si Konsehal Ariel Anghay, Wilfredo Bacareza, Bienvenido Badelles, Leo Pairat, Ronaldo Espina ug Konsehal Orlando Maglinao, pulos mga sakop sa LDP, nga dili nila pasagdan si Rovira sa iyang mga gimbuhaton isip bag-ong konsehal sa dakbayan.
Apan namahayag si Rovira nga dili sa partido nga kaayuhan ang gilantaw niining iyang pagkatudlo pagka-konsehal kondili ang kaayuhan sa dakbayan sa Iligan.
Sa usa ka interview sa telebisyon (ABS-CBN) si Rovira nagkanayon nga motabang siya sa pagpaningkamot nila ni Mayor Franklin M. Quijano ug Presidente Gloria Macapagal-Arroyo nga mabuksan pagbalik ang nasiradong planta sa puthaw nga mao ang National Steel Corporation (NSC). Iyang giklaro nga dili siya mamulitika sa hawanan sa konseho tungod kay dili man siya modagan sa umaabot 2004 local ug national election.
Type
Set
Start
End
Fea
Person
NE
44
60
{rule=PersonFinal, rule1=PersonFull, gender=male}
Default annotations
NE annotations
Date
FirstPerson
Initials
JobTitle
Known
LocKey
Location
Lookup
Organization
Person
Sentence
SpaceToken
Split
Spur
Spurious
Coreference data
NE
Sangguniang Panlung
M. Quijano
Mayor Franklin
Leo Pairat

A closer look at Cebuano

Balitang Bisaya
Ni Michael Kundiman, Mindanao Scoop, 2 February 2003
Rovira gitudlo na nga puli ni Adeva sa SP

GIDAWAT na sa mayoriya sa konseho ang pagkatudlo ni Atty. Voltaire I. Rovira isip bag-ong konsehal sa dakbayan sa Iligan hulip sa gibiyaang posisyon ni kanhi konsehal-anhing Atty. Narciso ?Boy? Adeva Jr. nga namatay niadtong Marso 6, 2001.

Sumala pa sa mga konsehal nga miyembro sa ruling majority, ilang ihatag ang ilang hugot nga suporta ngadto kang Rovira aron ilang mahatag ang mga komitiba nga angayan niyang huptan didto sa konseho.

Si Rovira gi-rekomendar ni Laban ng Demokratikong Pilipino national president, Sen. Edgardo Angara, ngadto sa Malaca?ang aron mohulip sa nabakanteng posisyon sa konseho dinhi sa Iligan. Ang iyang appointment gitiman-an ni Executive Secretary Alberto Rumulo ug giaprobahan na sa Department of Interior and Local Government (DILG).

Evaluation (1)

- System annotated 10 news texts and output as colour-coded HTML.
- Evaluation on paper by native Cebuano speaker from University of Maryland.
- Evaluation not perfect due to lack of annotator training
- 85.1% Precision, 58.2% Recall, 71.7% Fmeasure
- Evaluation was non-reusable because we didn't have a gold standard ☹

Evaluation Results

Cebuano	P	R	F	Baseline	P	R	F
Person	71	**65**	**68**		**86**	36	36
Org	75	**71**	**73**		**81**	47	38
Location	**73**	**78**	**76**		65	7	12
Date	**83**	**100**	**92**		42	58	49
Total	**76**	**79**	**77.5**		45	41.7	43

What did we learn?

- Even the most bizarre (and simple) ideas are worth trying
- Trying a variety of different approaches from the outset is fundamental
- Good gazetteer lists can get you a long way
- Good mechanisms for evaluation need to be factored in

Conditional Processing

What is conditional processing?

- In GATE, you can set a processing resource in your application to run or not depending on certain circumstances
- You can have several different PRs loaded, and let the system automatically choose which one to run, for each document.
- This is very helpful when you have texts in multiple languages, or of different types, which might require different kinds of processing
- For example, if you have a mixture of German and English documents in your corpus, you might have some PRs which are language-dependent and some which are not
- You can set up the application to run the relevant PRs on the right documents automatically.

Conditional processing with different languages

- Suppose we have a corpus with documents in German and English, and we only want to process the English texts.
- First we must distinguish between the two kinds of text, using a language identification tool
- For this we can use TextCat, which is a GATE plugin
- We use this (in default mode) to add a feature on each document, telling us which language the document is in
- Then we run a conditional processing pipeline, that only runs the subsequent PRs if the value of the language feature on the document is English
- The other documents will not be processed

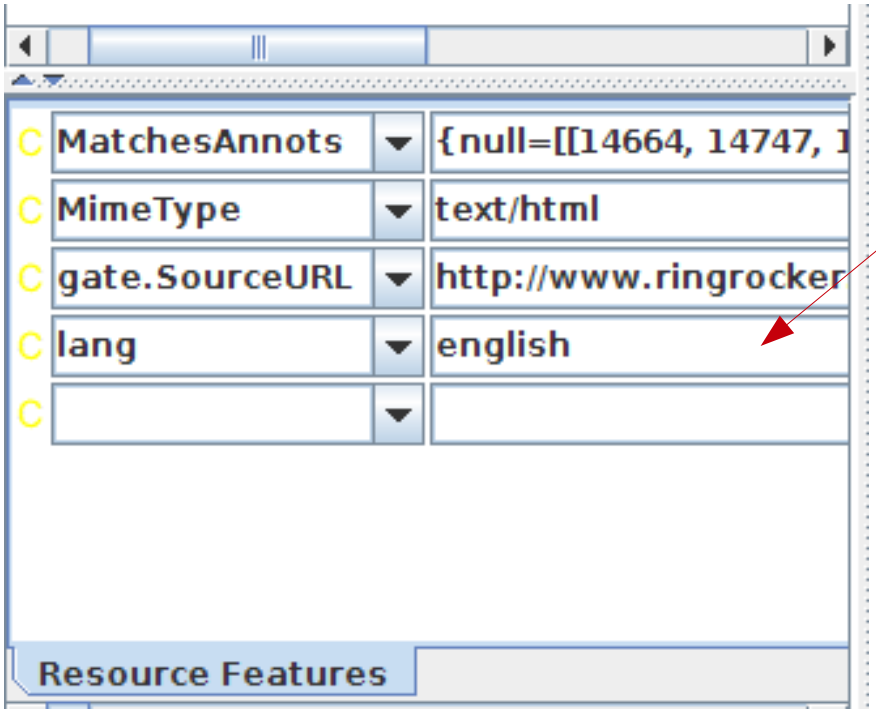
Hands-on with multilingual corpora (1)

- Create a new corpus in GATE and populate it with the two documents found in module-10-hands-on/corpus/rar-english-german-corpus/
- You should have one English and one German document loaded
- Load ANNIE and run it on the corpus
- You should get some sensible annotations for the English documentand some slightly less sensible ones for the German one

Hands-on with multilingual corpora (2)

- Use the GATE Plugins Manager to load the Language_Identification plugin
- Load the PR “TextCat Language Identification”
- Right click on ANNIE and select “Make pipeline conditional”
- You should now see a traffic light system in the application viewer, with all PRs currently set to green (Go!)
- Add the TextCat PR to the ANNIE pipeline directly after the Document Reset PR
- Save your application somewhere and run it on the corpus

Check the language of the documents

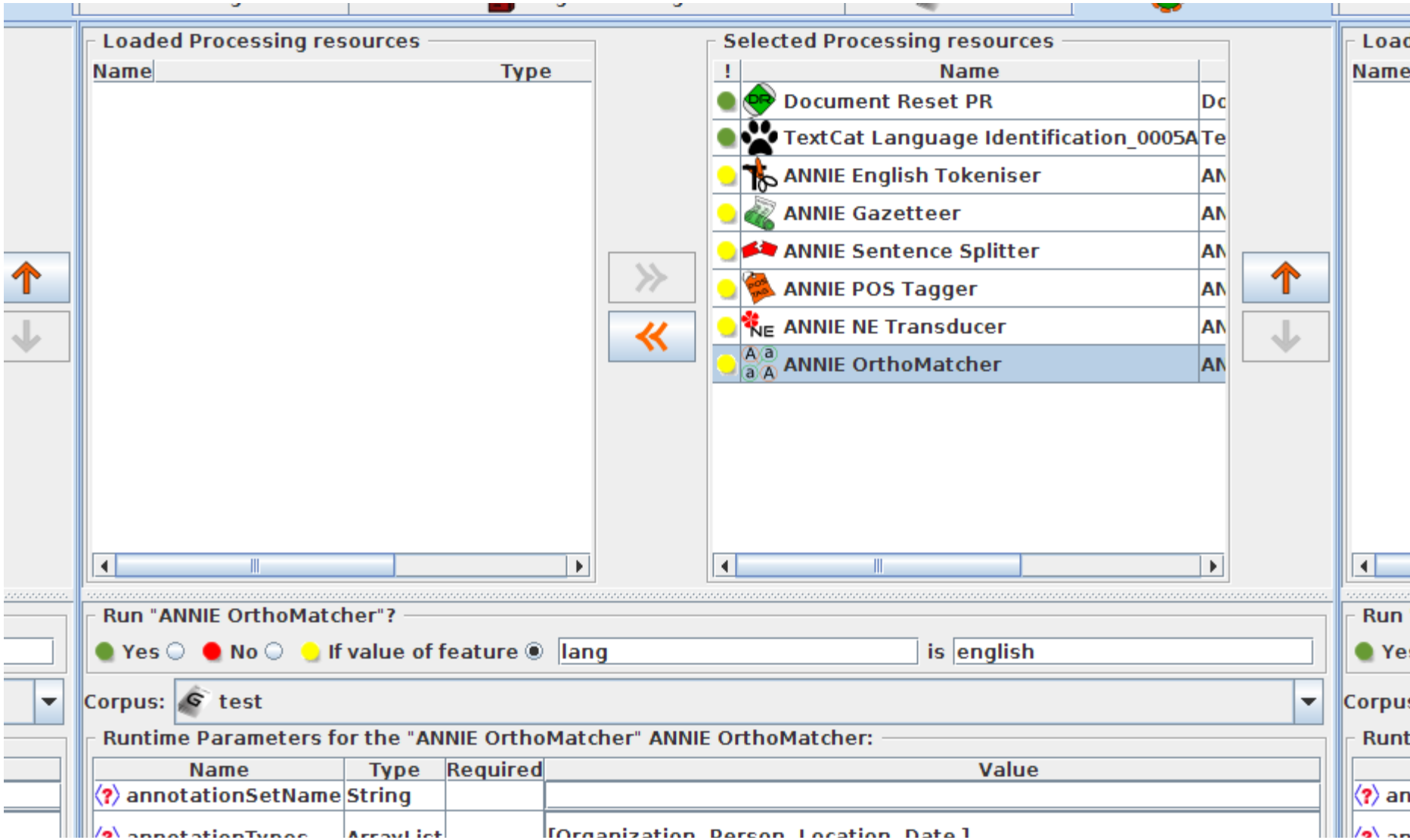


- Click on a document
- In the bottom left corner is the document features pane
- TextCat will add a language feature here

Run ANNIE conditionally

- Now let's make the PRs conditional
- We only want ANNIE to run on English documents, not German ones
- For each PR after the TextCat PR in the pipeline, click on the yellow dot and set it to run only if the value of the feature “lang” is “english”
- Save the application
- Run the application again and check both documents
- Only the English one should have annotations

Your application should look like this



What if we want to process the German too?

- If we want to process both German and English documents with differerent resources, we have a couple of options
  1. We can just call some language-specific PRs conditionally, and use the language-neutral PRs on all documents
  2. We can call differerent applications from within the main application
- The following two hands-on exercises demonstrate the difference between these

Hands-on with multilingual apps (1)

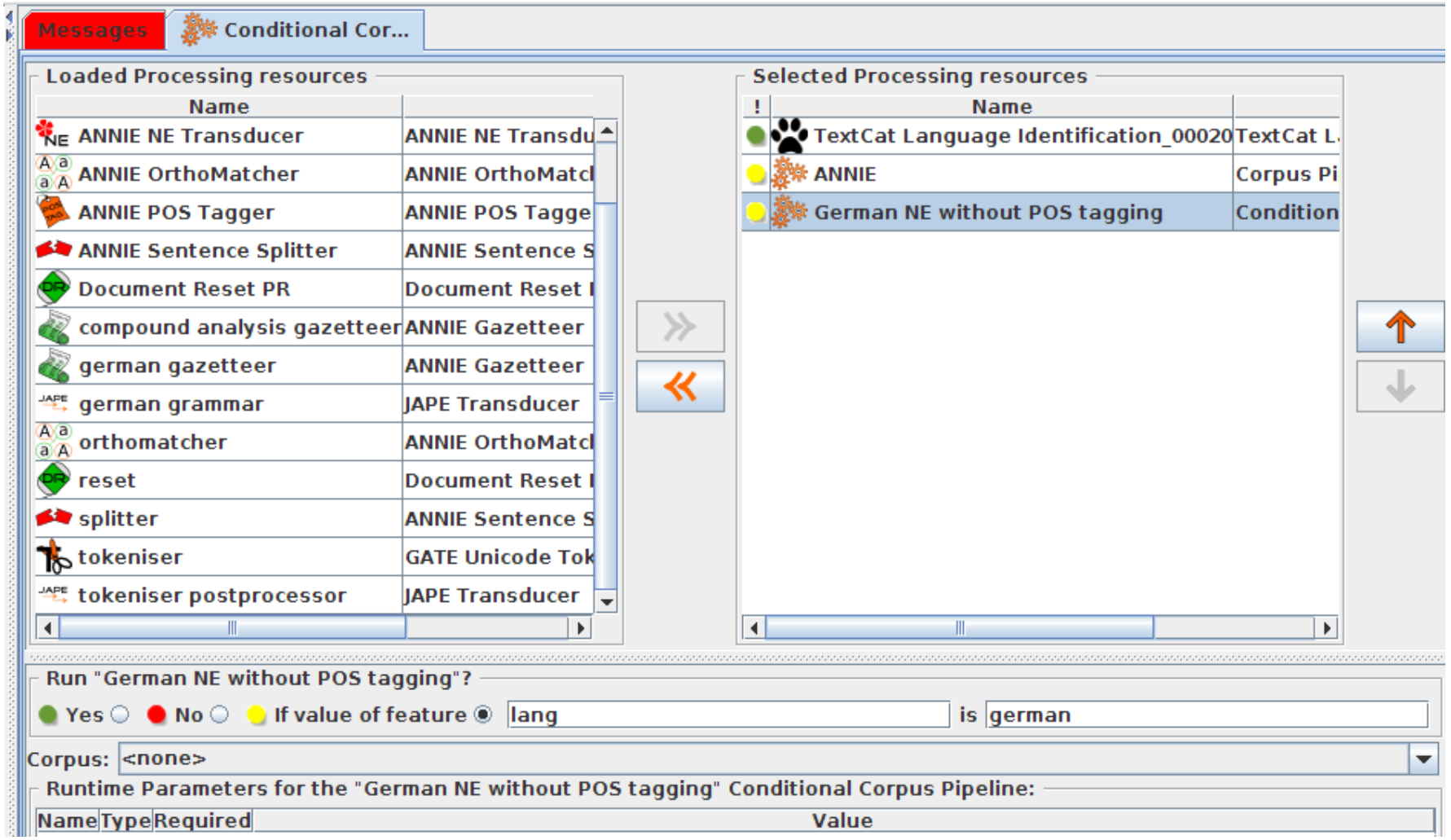
- Close all applications and resources in GATE (leave the corpus loaded)
- Load the application module-10-hands-on/annie+german.gapp
- Look at the various PRs in the app: some are set to run on English documents, some on German ones, and some on all documents
- Run the application on your corpus
- The German document should now be annotated with German NEs and the English document with English ones

Hands-on with multilingual apps (2)

- Close all applications and resources you have loaded in GATE
- Load ANNIE with defaults
- Load the German IE application from “Ready-made applications”
- Create a new conditional corpus pipeline
- Load a TextCat PR and add it to the new pipeline created
- Add the ANNIE and German **applications** to the pipeline (in either order) after the TextCat

  Set ANNIE to run on English documents and the German app to run on German ones
- Save the main application and run it on your corpus

Your application should look like this



Conditional Processing for Different Genres

- Let's take the example of texts in different domains: a text about sport might require some different grammar rules

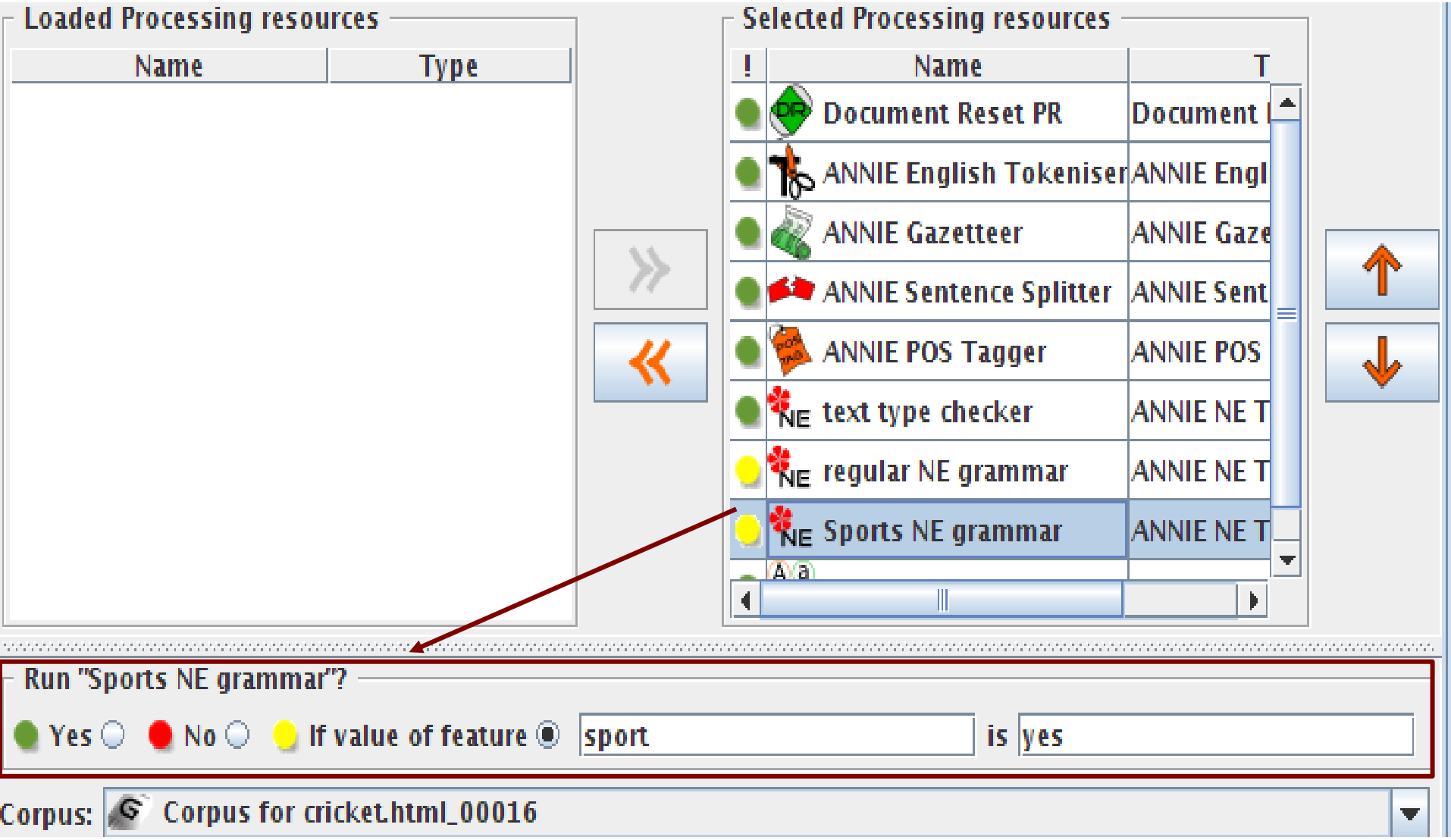
"Michael Di Venuto and Kyle Coetzer both hit centuries as Durham piled on the runs to take early charge of the season curtain-raiser against the MCC."

- Here "Durham" is an Organisation (the Durham cricket team) not a Location (or Person).
- If you have a corpus of news texts, you might want to separate the sports texts from the non-sports ones, so that you can process them differently

How does it work?

- First we must distinguish between the different texts, and annotate them with different values for a document feature
- Use a JAPE grammar to find texts about sport, e.g. by recognising sports words in the text from a gazetteer
- JAPE grammar adds a document feature “sport” with value “yes” to sports documents, and with value “no” to other documents
- Use a **conditional** corpus pipeline rather than a normal corpus pipeline to create the application
- Add both the regular grammar and the sports grammar to the application
- Set the sports grammar to run only if the value of the feature “sport” is “yes”
- Set the regular grammar to run only if the value is “no”

Running PRs conditionally

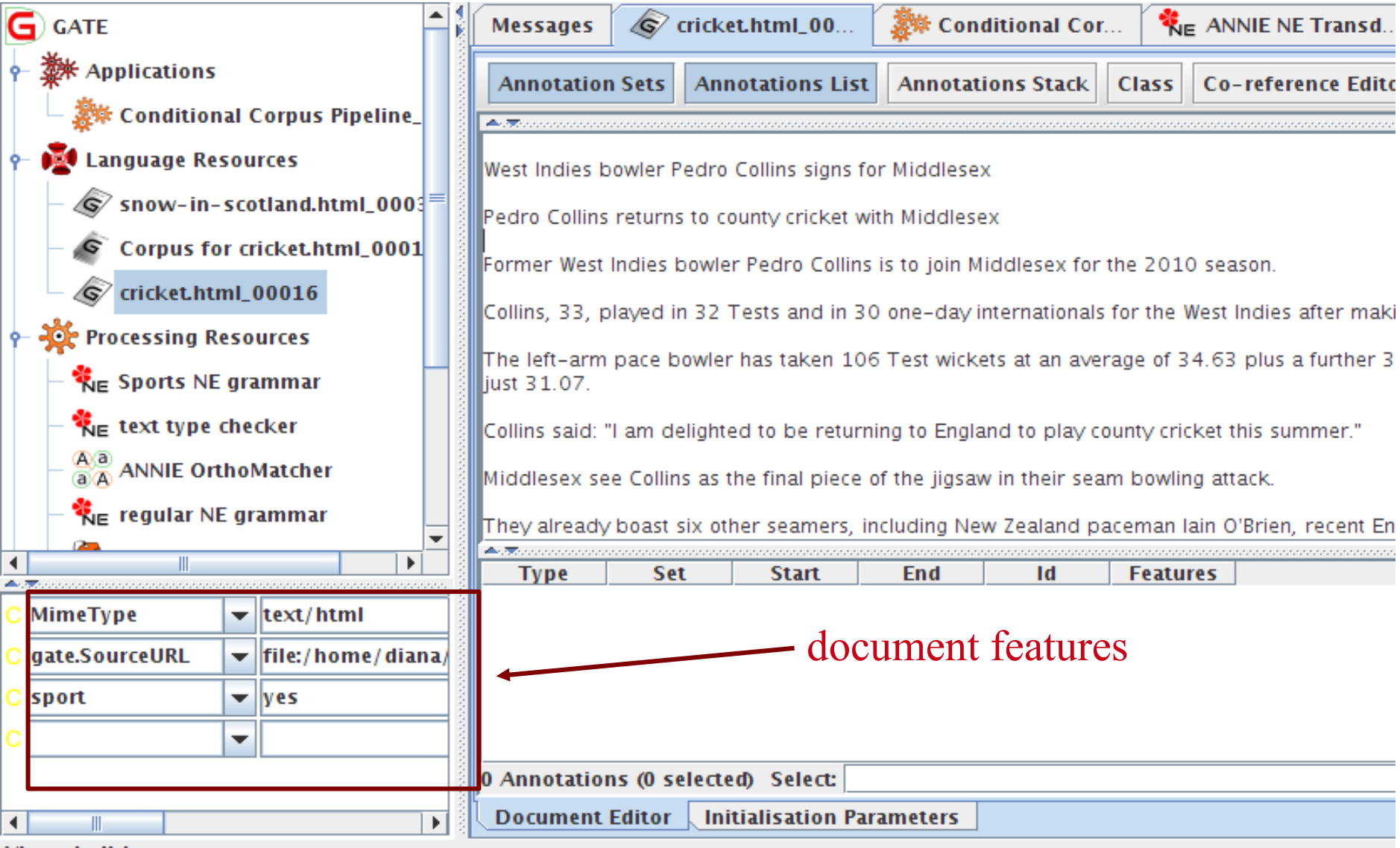


Setting document features

- Just like creating an annotation on a piece of text, you can also create features and values on the whole document, both manually and automatically
- Rule to annotate a document with feature “sport” and value “yes” if it contains any sports-related words

```
Rule: AnnotateWithSport
(
{Lookup.majorType == sport}
)
-->
 {
 doc.getFeatures().put("sport", "yes");
 }
```

Viewing document features



Resetting features

- Unlike regular annotations, document features are not removed by the Document Reset PR
- The only way to remove a document feature is either manually, or using another JAPE rule to remove the feature or change its value
- We could remove the sport feature with the following piece of code

```
{
doc.getFeatures().remove("sport”);
}
```

Setting the feature for non-sports texts

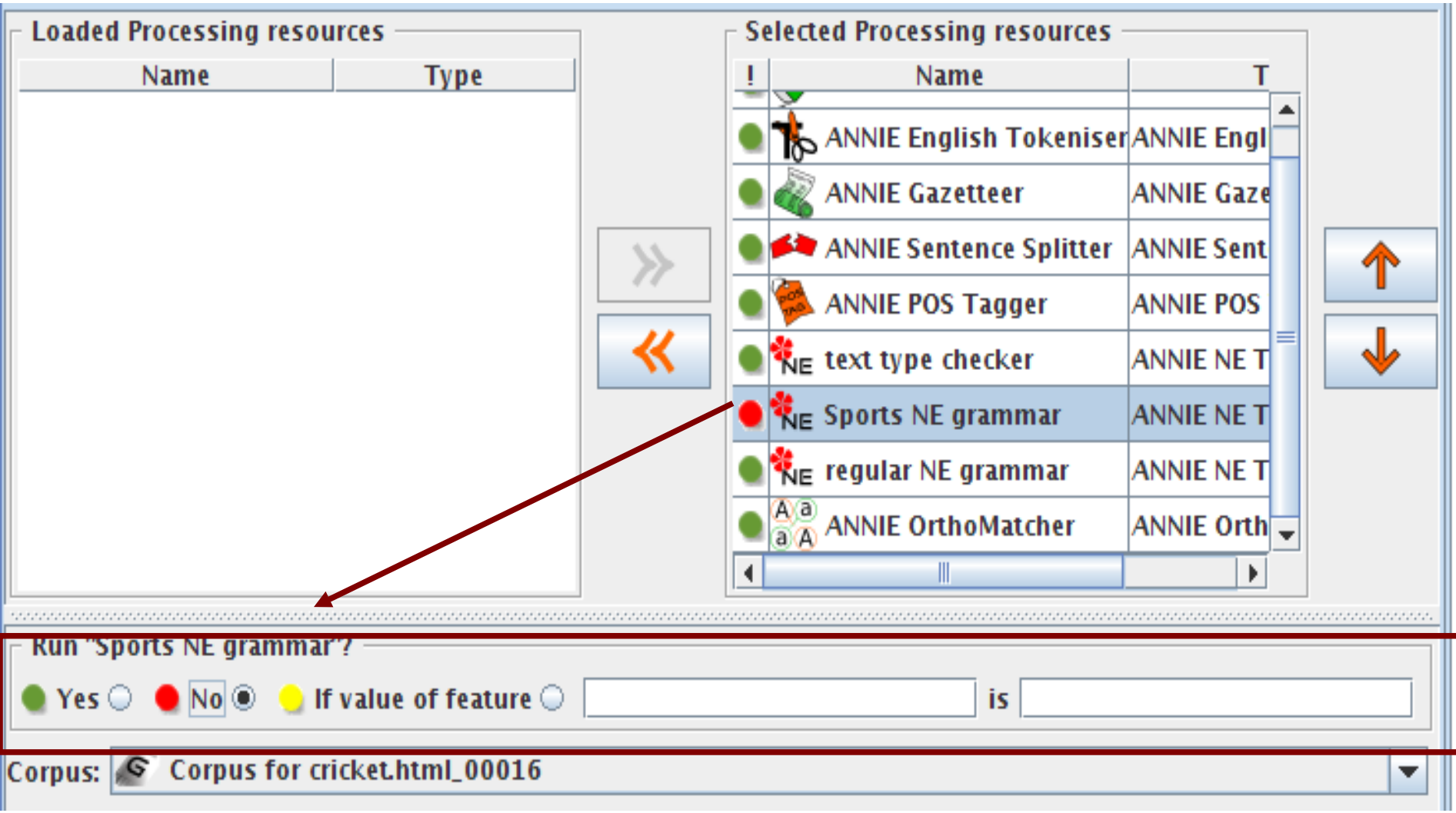
- How do we now annotate all non-sports texts with the “no” value?
- The easiest way is to first annotate ALL texts with this value.
- When we then run the sports grammar, it will replace this with the “yes” value for any documents that meet the constraints
- We therefore add a previous grammar phase which annotates all non-empty documents with “sport = no”
- We use the “once” matching phase so that the grammar exits as soon as the first Token has been found

```
Rule: AnnotateAll
({Token})
-->
 {  doc.getFeatures().put("sport", "no");  }
```

Other ways to use conditional processing

- You can also set a PR to just not run at all, within an application
- The usual reason for this is for testing purposes
- When you remove a PR from the application, you may forget the order in which you had PRs set, or you may even forget which PRs were in the application
- If you remove the PRs from GATE, you may also lose the runtime settings you had associated with them
- It's easier to just set the PR not to run by clicking the red button
- You can save the application with the PR set to not run, and then you (and other people) can easily change this when you reload the application

Setting a PR not to run



Extra Hands-on Exercise: Genre

- Load the application hands-on/conditional-sports.gapp
- It should load 2 texts automatically
- Run the application on the corpus and look at the results for Location and Organization
- Try turning on and off the JAPE grammars that set the document features for sport (document-sport grammar), and look at the resulting value of the document features in each case
- Turn off the sports grammar and set the main ANNIE grammar to run on the cricket document. See the difference in the Organization and Location annotations
- Try turning on and off other PRs as you want, or try editing the document features manually.

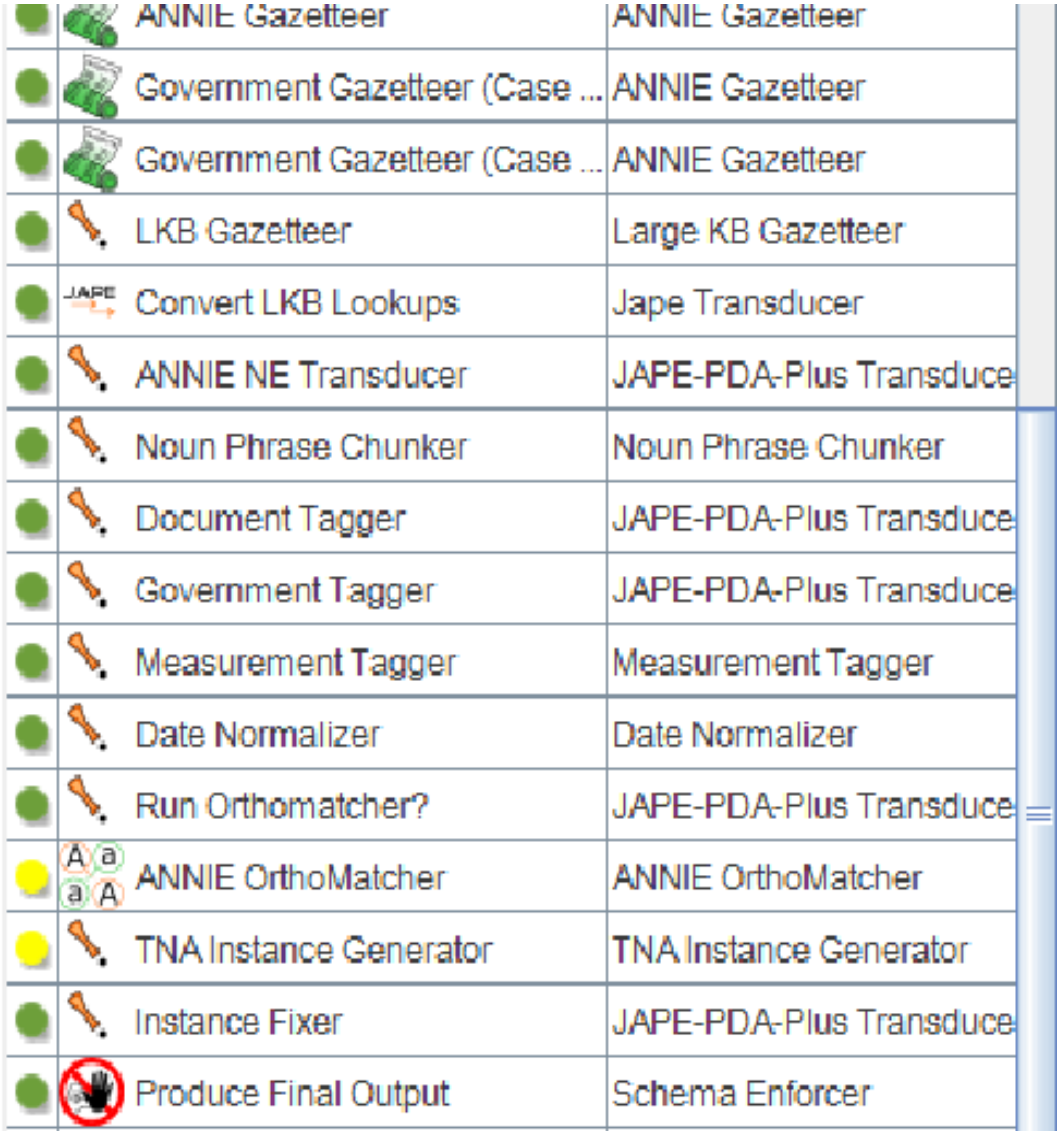
Other uses for conditional processing

- Processing degraded text along with normal text
- For degraded text (e.g. emails, ASR transcriptions) you might want to use some case-insensitive PRs
- Another use is in combination with different kinds of files (HTML, plain text etc) which might require different processing
- More about this later....

Another example

- In one application we developed, we found a problem when running the Orthomatcher (co-reference) on certain texts where there were a lot of annotations of the same type.
- To solve this issue, we first checked to see how many annotations of each were present in a document
- If more than a certain number were present, we added a document feature indicating this
- We then set the orthomatcher to only run on a document which did not contain this feature.

Application



Grammar to check number of annotations

If there are more than 200 annotations of one type, don't run the orthomatcher

```
Rule: CheckAnnotations
({Person}|{Organization}|{Location})
-->
{
AnnotationSet annots = inputAS.get("Person");
if (annots.size() > 200) {
doc.getFeatures().put("runOrthomatcher","false");
return;}
...
doc.getFeatures().put("runOrthomatcher","true");
}
```

Section by Section Processing: the Segment Processing PR

What is it?

- PR which enables you to process labelled sections of a document independently, one at a time
- Useful for
  - very large documents
  - when you want annotations in different sections to be independent of each other
  - when you only want to process certain sections within a document

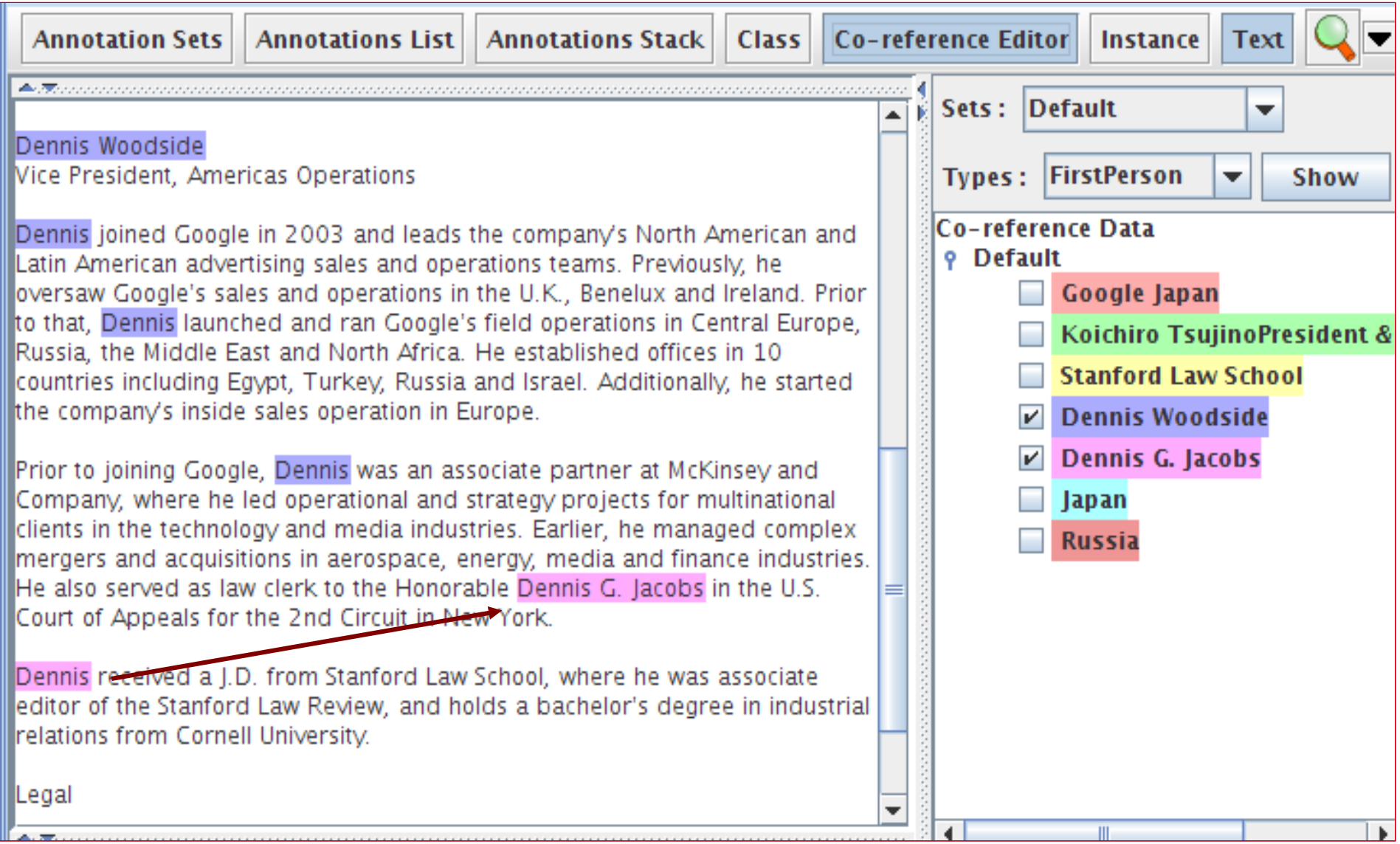
Processing large documents

- If you have a very large document, processing it may be very slow
- One solution is to chop it up into smaller documents and process each one separately, using a datastore to avoid keeping all the documents in memory at once
- But this means you then need to merge all the documents back afterwards
- The Segment Processing PR does this all in one go, by processing each labelled section separately
- This is quicker than processing the whole document in one go, because storing a lot of annotations (even if they are not being accessed) slows down the processing

Processing Sections Independently

- Another problem with large documents can arise when you want to handle each section separately
- You may not want annotations to be co-referenced across sections, for instance if a web page has profiles of different people with similar names
- Using the Segment Processing PR enables you to handle each section separately, without breaking up the document
- It also enables you to use different PRs for each section, using a conditional controller
- For example, some documents may have sections in different languages

Problematic co-references



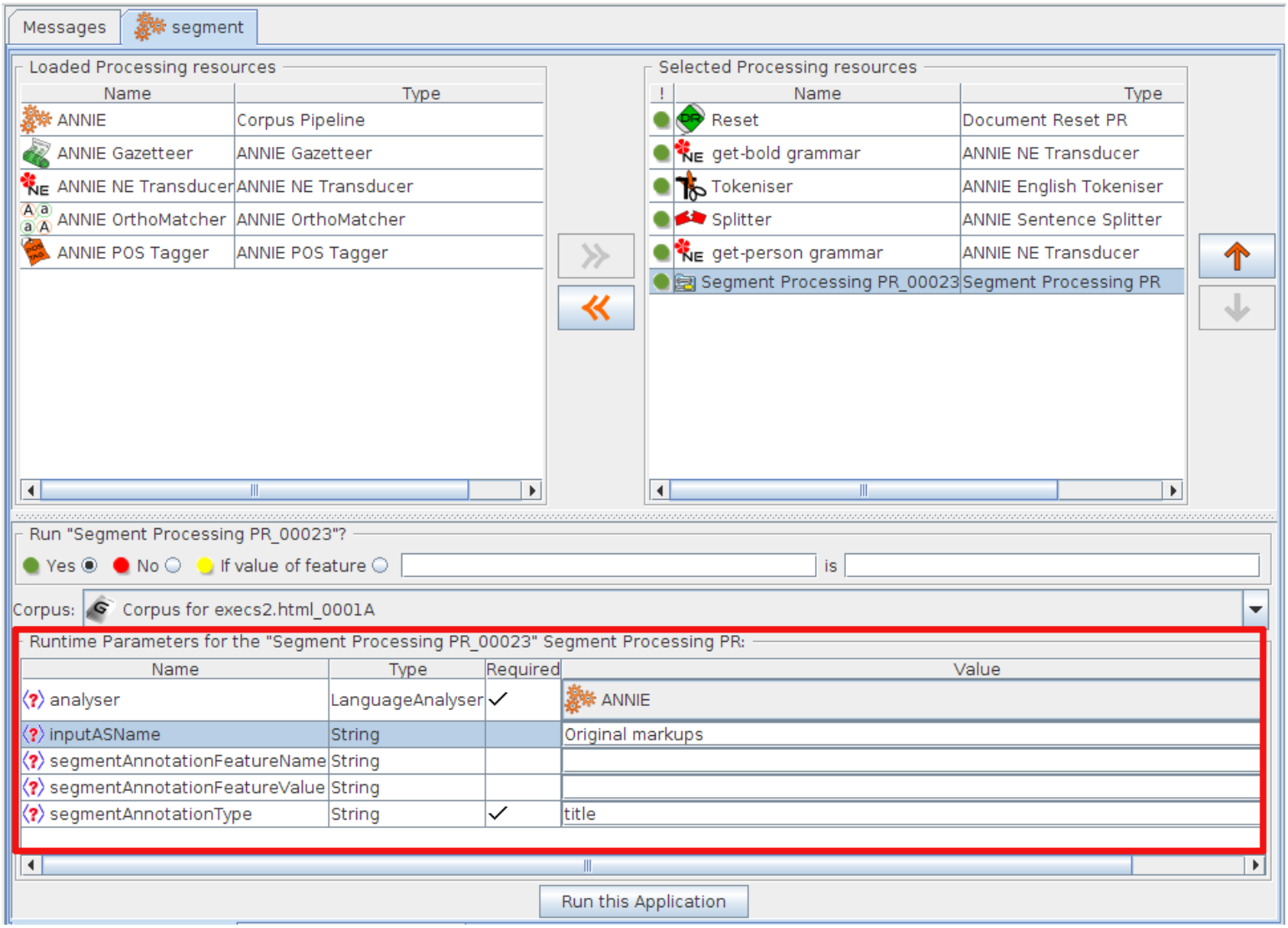
Getting rid of the junk

- Another very common problem is that some documents contain lots of "junk" that you don't want to process, e.g. HTML files contain javascript or contents lists, footers etc.
- There are a number of ways in which you can do this: you may need to experiment to find the best solution for each case
  - **Segment Processing** PR enables you to only process the section(s) you are interested in and ignore the junk
  - Using the **AnnotationSetTransfer** PR, though this works slightly differently
  - Using the **Boilerpipe** PR - this works best for ignoring standard kinds of boilerplate

How does it work?

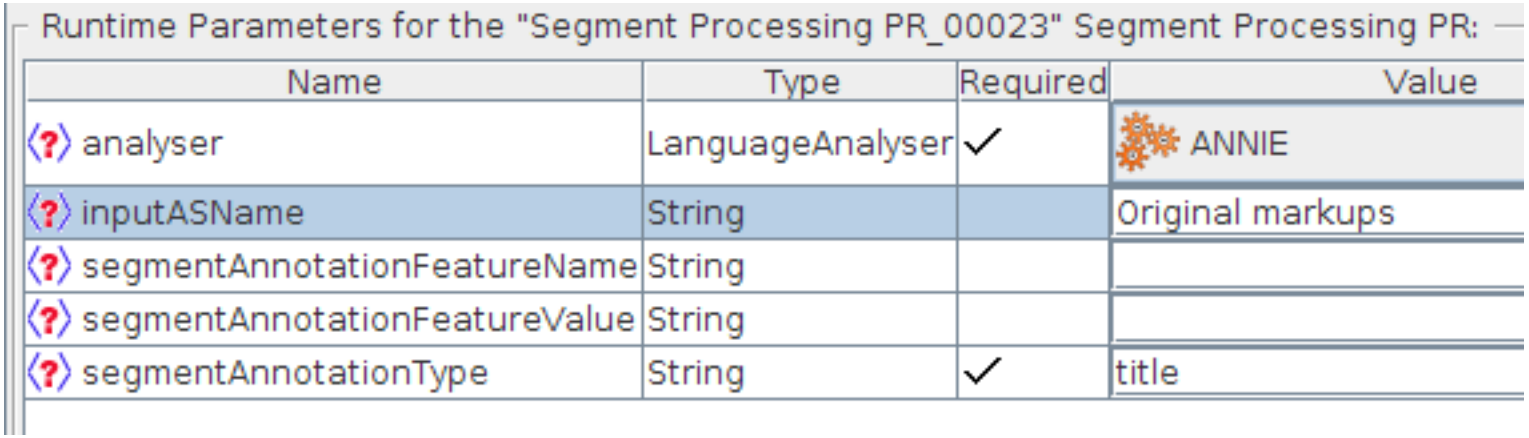
- The PR is part of the Alignment Plugin
- A new application needs to be created, containing the Segment PR
- The PR then takes as one of its parameters, an instance of the application that you want to run on the document (e.g. ANNIE)
- You can add other PRs before or after the Segment PR, if you want them to run over the whole document rather than the specified section(s)

Running ANNIE on a title segment



- Application contains a Segment Processing PR
- Segment Processing PR calls ANNIE application

Segment Processing Parameters

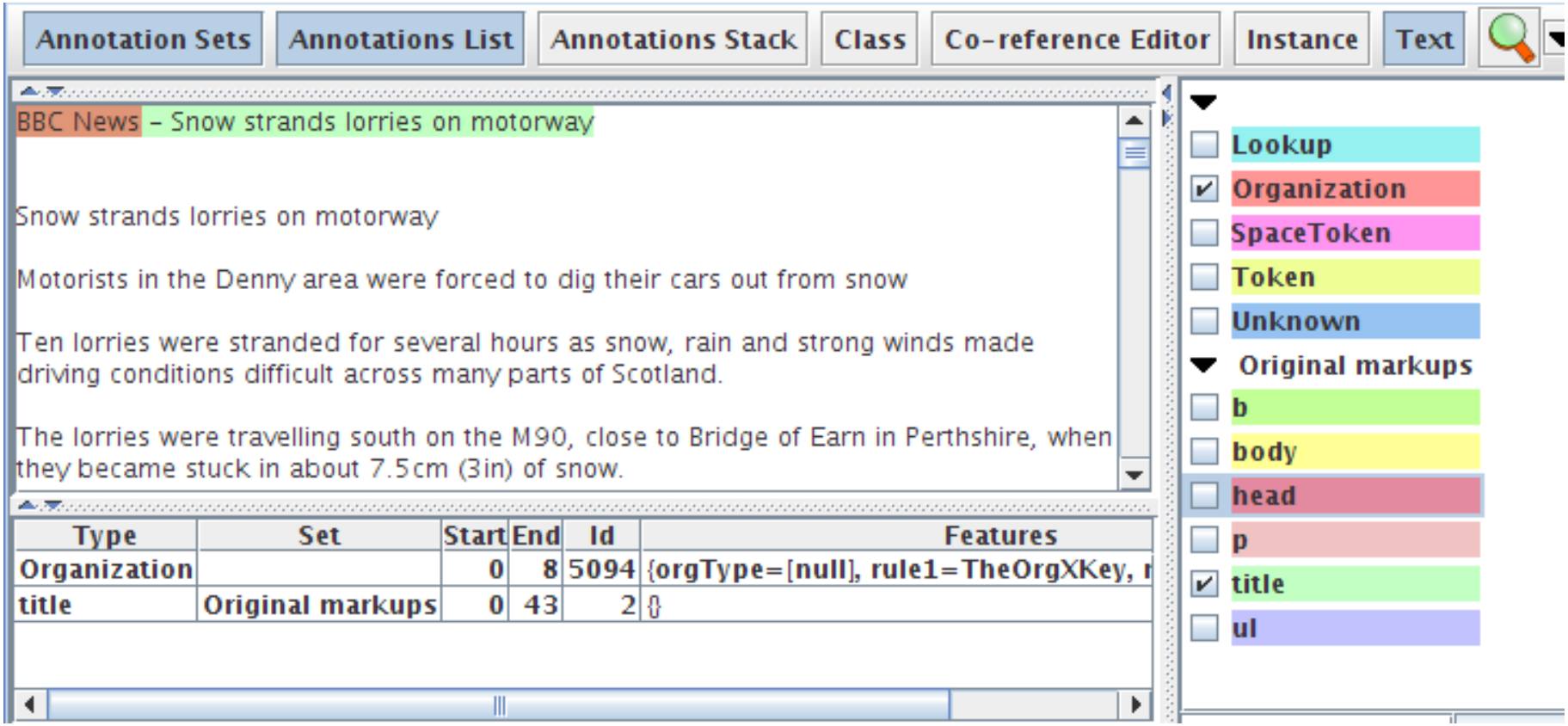

Runtime Parameters for the "Segment Processing PR_00023" Segment Processing PR:

Name	Type	Required	Value
analyser	LanguageAnalyser	✓	ANNIE
inputASName	String		Original markups
segmentAnnotationFeatureName	String		
segmentAnnotationFeatureValue	String		
segmentAnnotationType	String	✓	title

- Segment Processing PR calls the ANNIE application

- ANNIE is set to run only on the text covered by the span of the "title" annotation in the Original markups annotation set

Annotation Result



- Green shading shows the title, which spans the section to be annotated
- The only NE found is the Organization “BBC News” in the title
- Tokens in the rest of the document are not annotated

Hands-on segment processing (1)

- Clear GATE of all PRs, applications and resources
- Load the application segment-processing.gapp
- Load the document execs2.html and add it to a corpus
- Run the application on the corpus
- This application first creates an annotation type “bold” in the default annotation set, using the “b” annotations from the Original markups set.
- Have a look at the grammar get-bold.jape and the runtime parameters for it to see how it was done.
- Then the application uses the get-person.jape grammar to match a bold annotation followed by a set of sentences, creating a new annotation “Content” in the default annotation set.
- Have a look at the “bold” and “Content” annotations in the document.

Hands-on segment processing (2)

- Now we have our document separated into sections by means of the Content annotation
- Load ANNIE with defaults. Remove the Document Reset, Tokeniser and Sentence Splitter from it (make sure you remove the ones named ANNIE Tokeniser, etc. and not the ones previously loaded)
- Create a Segment Processing PR and add it to the end of your Segment application.
- Select the Segment Processing PR in the application and set the “analyser” value to “ANNIE”
- Set the value of “segmentAnnotationType” to “Content”
- Run the application and look at the results
- Look at the co-references created: they should not cross Content boundaries. Look at “Google” annotations for an example.

Using multiple annotation sets

Annotation Set Transfer

- This PR enables copying or moving annotations from one set to another
- As with the Segment Processing PR, you can specify a covering annotation to delimit the section you're interested in
- One use for this is to change annotation set names or to move results into a new set, without rerunning the application
- For example, you might want to move all the gold standard annotations from Default to Key annotation set

Transferring annotations

The annotations remain the same, they're just stored in a different set

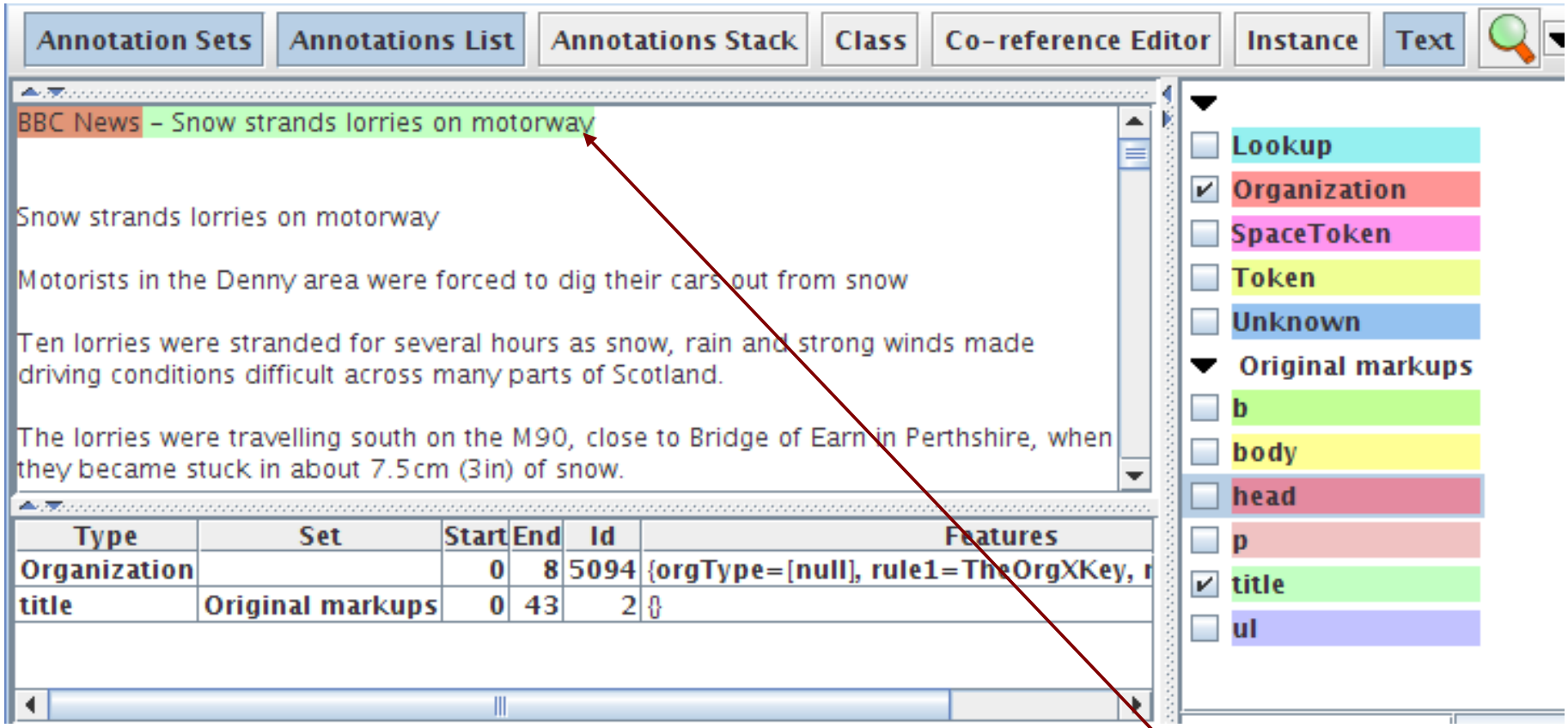
Hands-on Exercise

- Scenario: You have asked someone to annotate your documents manually, but you forgot to tell them to put the annotations in the Key set and they are in the Default set
- Clear GATE of all previous documents, corpora, applications and PRs
- Load document self-shearing-sheep-marked.xml and create an instance of an AST (you may need to load the Tools plugin)
- Have a look at the annotations in the document
- Add the AST to a new application and set the parameters to move all annotations from Default to Key
- Make sure you don't leave the originals in Default!
- Run the application and check the results

Delimiting a section of text

- Another use is to delimit only a certain section of text in which to run further PRs over
- Unlike with the Segment Processing PR, if we are dealing with multiple sections within a document, these will not be processed independently
- So co-references will still hold between different sections
- Also, those PRs which do not consider specific annotations as input (e.g. tokeniser and gazetteer), will run over the whole document regardless

Processing a single section

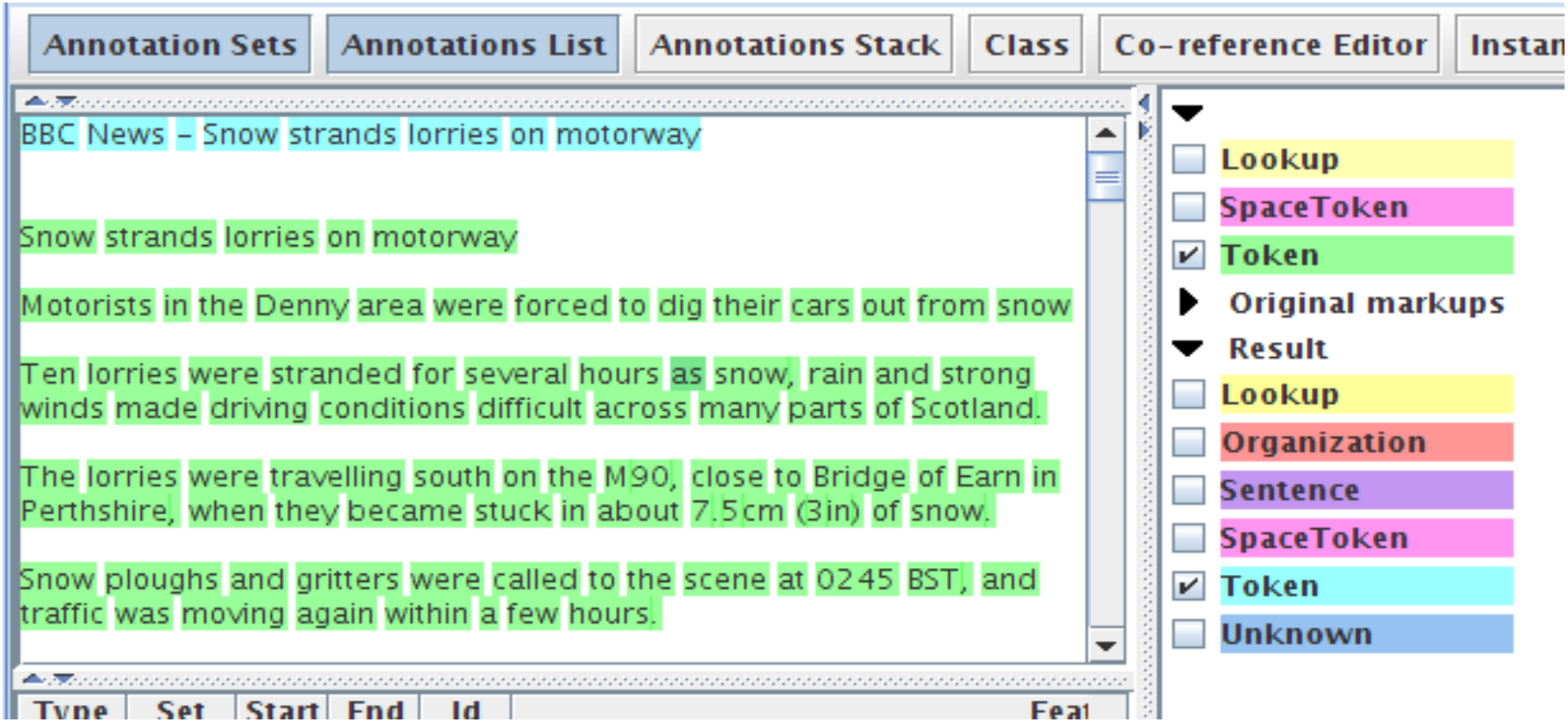


- Only the “title” section is annotated with NEs

title

Transferring title annotations

- But the rest of the document remains tokenised
- These Tokens remain in the Default set because they weren't moved.



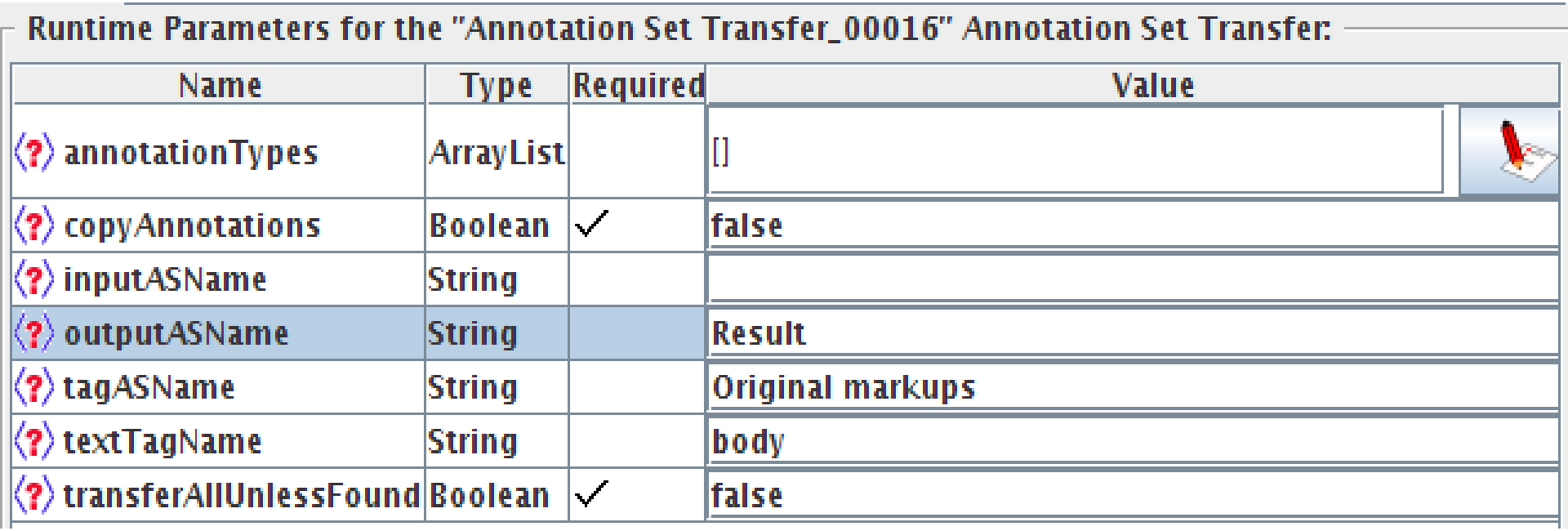
Setting the parameters

- Let's assume we want to process only those annotations covered by the HTML “body” annotation (i.e. we don't want to process the headers etc.).
- We'll put our final annotations in the “Result” set.
- We need to specify as parameters
  - **textTagName**: the name of the covering annotation: “body”
  - **tagASname**: the annotation set where we find this: “Original markups”
  - **annotationTypes**: which annotations we want to transfer
  - **inputASname**: which annotation set we want to transfer them from: “Default”
  - **outputASname**: which annotation set we want to transfer them into: “Result”

Additional options

- There are two additional options you can choose
  - **copyAnnotations**: whether to copy or move the annotations (i.e. keep the originals or delete them)
  - **transferAllUnlessFound**: if the covering annotation is not found, just transfer all annotations. This is a useful option if you just want to transfer all annotations in a document without worrying about a covering annotation.

Parameter settings



Runtime Parameters for the "Annotation Set Transfer_00016" Annotation Set Transfer:

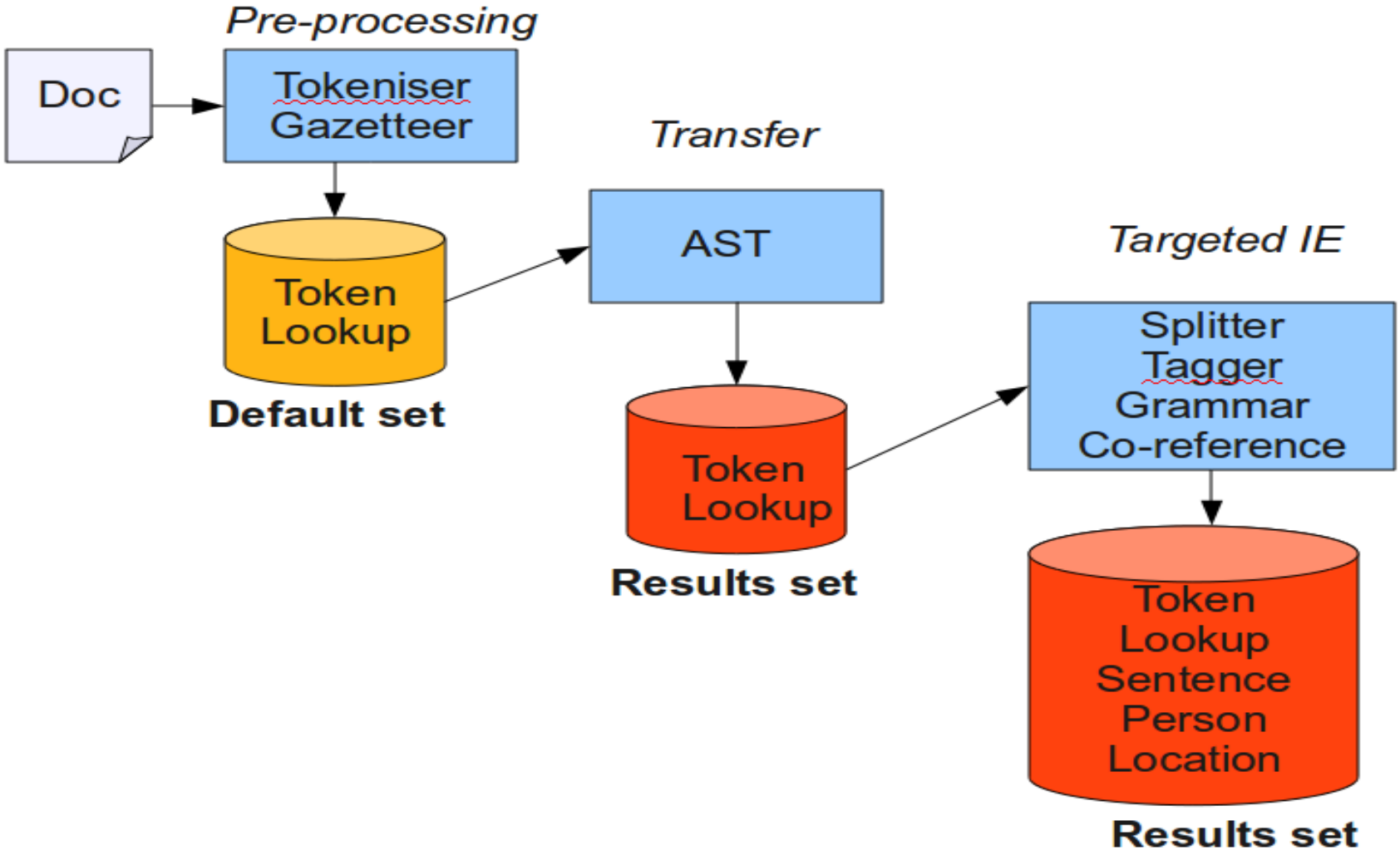
Name	Type	Required	Value
annotationTypes	ArrayList		[]
copyAnnotations	Boolean	✓	false
inputASName	String		
outputASName	String		Result
tagASName	String		Original markups
textTagName	String		body
transferAllUnlessFound	Boolean	✓	false

- Move all annotations contained within the “body” annotation (found in the Original markups set), from the Default set to the Result set.
- If no “body” annotation is found, do nothing.

Using it within an application

- We want to run ANNIE over only the text contained within the “body” text
- Apart from the tokeniser and gazetteer, the other ANNIE PRs all rely on previous annotations (Token, Lookup, Sentence, etc.)
- We run the tokeniser and gazetteer first on the whole document
- Then use the AST to transfer all relevant Token and Lookup annotations into the new set
- Then we can run the rest of the ANNIE PRs on these annotations
- To do this, we use for inputAS and outputAS the name of the new set “Result”

Application architecture



Hands-on: processing a document section

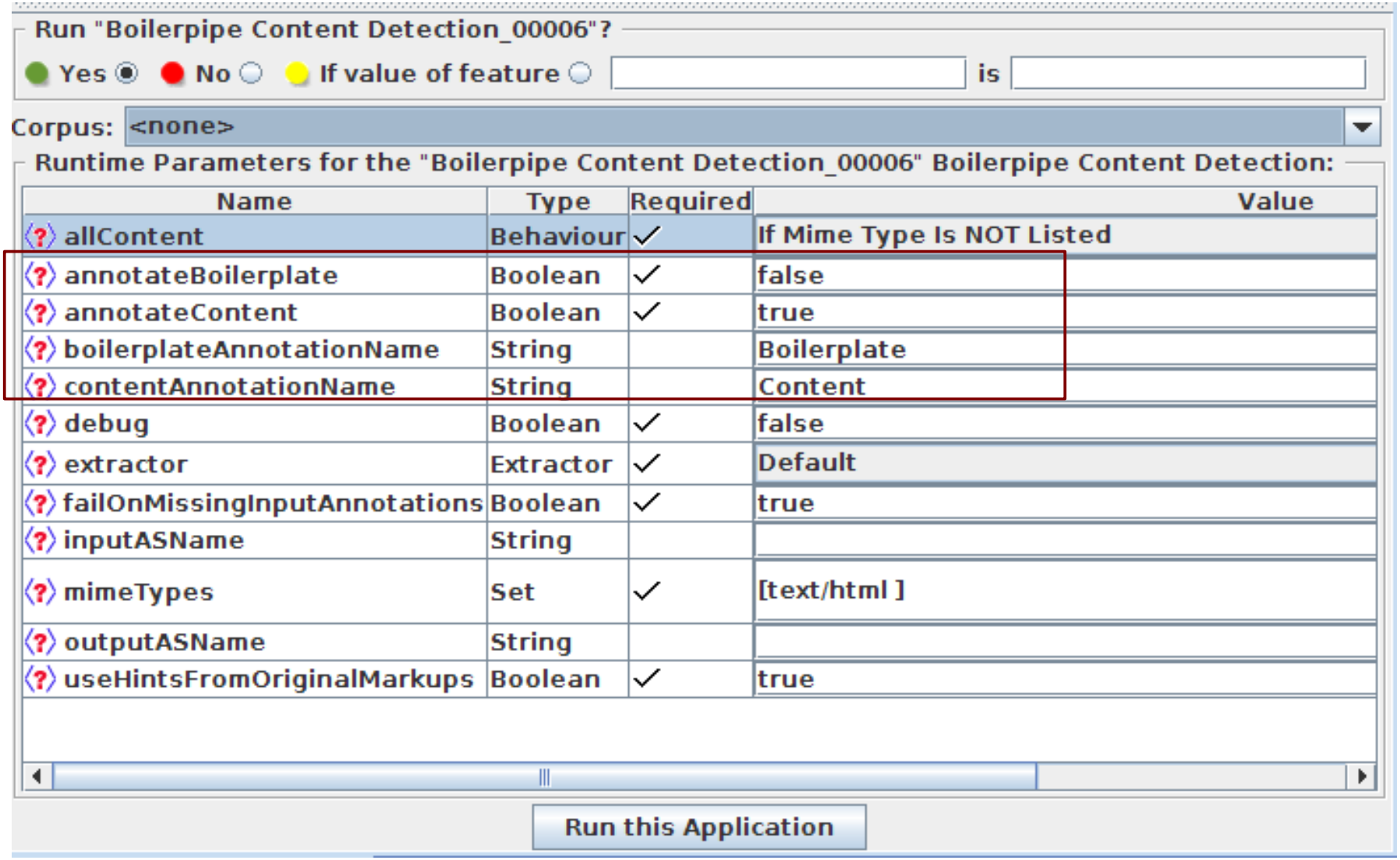
- Modify ANNIE to only run over the body of the document
- Load ANNIE with defaults
- Add the AST immediately after the tokeniser and gazetteer
- Set the AST parameters to move all annotations contained within the “body” annotation (found in the Original markups set), from the Default set to the Result set.
- If you get stuck, check the slide “Setting the Parameters”
- Modify the Input and Output set of all following PRs to “Result”
- Run on the corpus and inspect the results
- Now try moving the AST to immediately before the tokeniser and gazetteer - what happens when you run it?

Content Detection using Boilerpipe

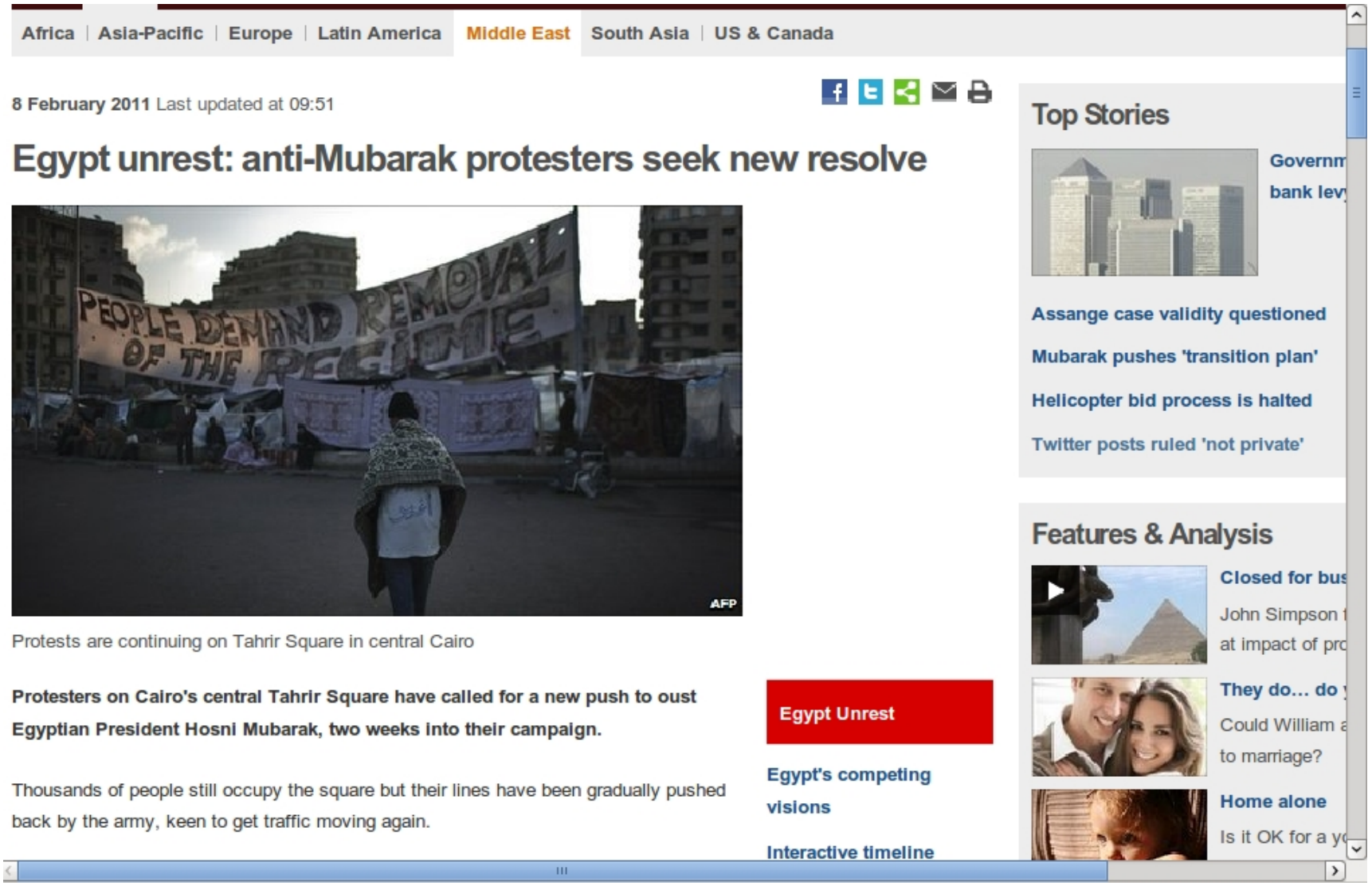
What is the Boilerpipe PR?

- In a closed domain, you can often write some JAPE rules to separate real document content from headers, footers, menus etc.
- In many cases, or when dealing with texts of different kinds or in different formats, it can get much trickier
- Boilerpipe PR provides algorithms to separate the surplus “clutter” (boilerplate, templates) from the main textual content of a web page.
- Applies the Boilerpipe Library to a GATE document in order to annotate the content, the boilerpipe, or both.
- Due to the way in which the library works, not all features from the library are currently available through the GATE PR

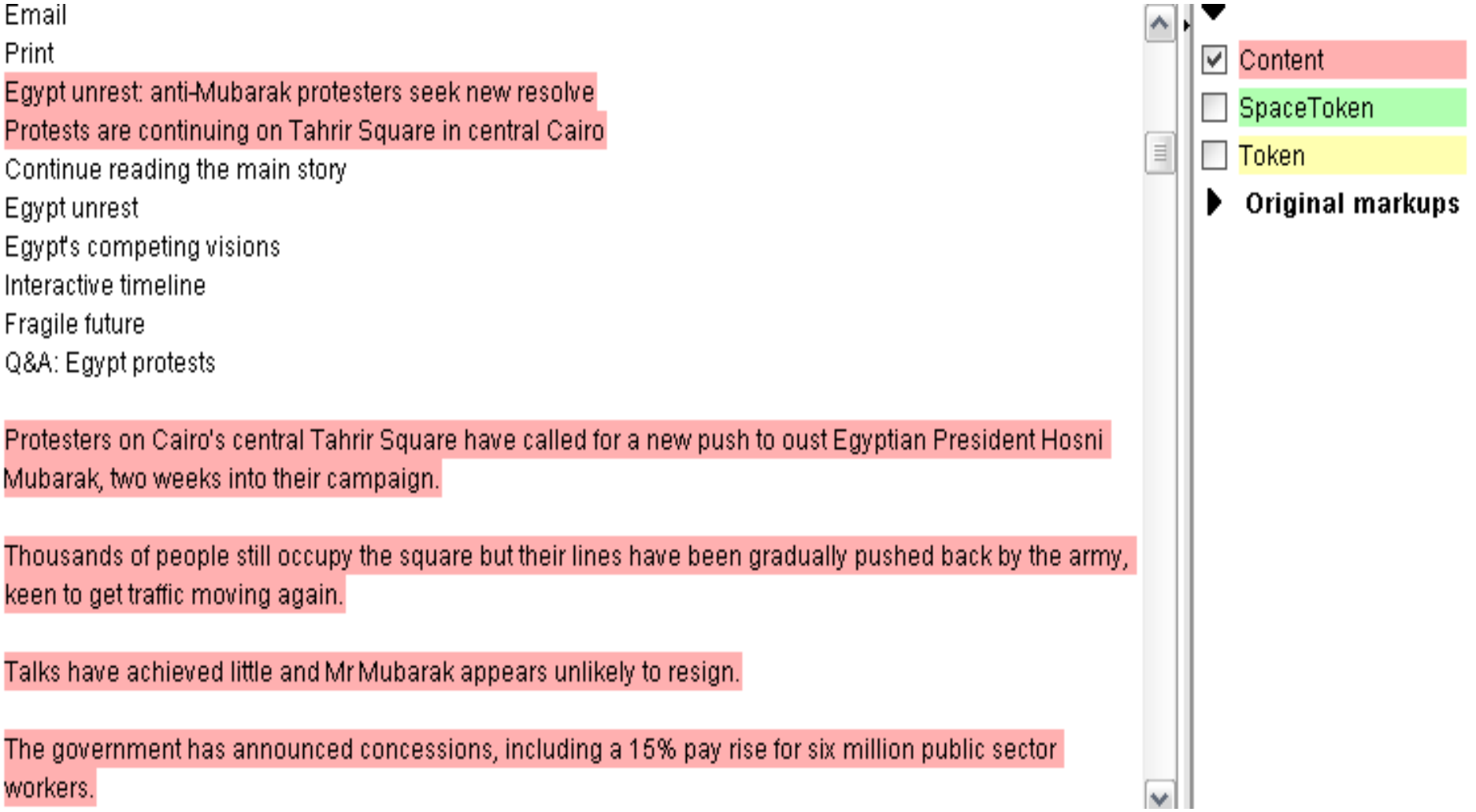
Boilerpipe Parameters



Original HTML document



Processed Document



Try it yourself

- Load the Tagger_Boilerpipe plugin
- Create a Boilerpipe Content Detection PR
- Create a new application, and add to it a Document Reset, a Tokeniser, and the Boilerpipe PR
- Leave all the parameters as default
- Load a document from the web, e.g. one of the pages from http://bbc.co.uk/news, add to a corpus, and run the application
- View the “Content” annotations on the document (in the Default set)
- Change the annotateBoilerplate parameter from false to true and rerun the application
- View the “Boilerplate” annotations

Schema Enforcer

- When creating an application, you often end up with lots of annotations and features which are not needed in the final output
- If pushing the output into a MIMIR index, or if storage space is an issue, it's particularly important to get rid of these
- You can tidy up the output using the AnnotationSetTransfer PR to move selected annotation types to a new set, but there's still the problem of the features
- Schema Enforcer PR will ensure that annotations and features will only appear in the final output set if they adhere strictly to the annotation schemas used
- Straightforward to use - load Schema Tools plugin and just list the schemas to be used in the runtime parameters (they must be loaded in GATE already)

The Groovy PR

Groovy Scripting PR

- Groovy is a dynamic programming language based on Java.
  - http://groovy.codehaus.org/
- The GATE Groovy plugin provides a powerful scripting PR that can be included in a corpus pipeline and run over each document.
- The script has full access to the current document and corpus through the GATE API, like a Java JAPE RHS but more powerful
- Unlike a JAPE Transducer, this PR does not have to match anything in the document in order to “fire the rules”

Groovy Scripting PR

- Two init parameters:
  - **scriptURL**, the path to the script
  - **encoding** (default UTF-8)
- Once the PR is created, the path to the file cannot be changed
- Just like JAPE, you can edit the file outside of GATE, save it, and re-initialize the PR to reload the file (and get syntax error messages)
- Three runtime parameters:
  - **inputASName** and **outputASName** (annotation sets)
  - **scriptParams** (key-value pairs)

Groovy Scripting PR

- Inside the script, you get 6 automatic variables “free of charge”:
  - **doc**, the current document (as in JAPE)
  - **corpus**, the current corpus
  - **content**, the string content of this document
  - **inputAS** and **outputAS**, the annotation sets for the current document named in the runtime parameters (as in JAPE)
  - **scriptParams**, a FeatureMap with the keys and values from the scriptParams runtime parameter, which lets you pass your own simple configuration options to the PR and change them from the pipeline interface without editing the script

Groovy Scripting PR

- What can you do with it?
  - Anything you can do in a JAPE Java RHS, and more
  - Read/write access to the document (features, content, all annotation sets)
  - Read/write access to the corpus (features, size, contents) but be careful
  - Control structures (loops, if then else, etc.)
  - No need to match a pattern of annotations
- Example: check each document for certain things and set its features accordingly
  - features can be used to regulate conditional PRs later in a conditional corpus pipeline, for example

Hands-on: Groovy Scripting PR

- Remove all existing documents, corpora, resources and applications from GATE
- Create a new corpus and populate it from **corpus-benchmark/test-corpus/clean** in the hands-on materials
- Load the ANNIE application and the Groovy plugin
- Create a new Groovy Scripting PR from the file **groovy/Example.groovy** in the hands-on materials, and add it to end of the ANNIE pipeline.

Groovy Scripting PR

```
// Get all the Person annotations
AnnotationSet persons = inputAS.get("Person");

// Print the name of the current document
println(doc.getName());

// Print the text underlying each Person annotation
for (person in persons) {
  println("   " + gate.Utils.stringFor(doc, person));
}

// Record the number of Person annotations
doc.getFeatures().put("nbr_persons", persons.size());

// Flag whether the document contains any Person annotations;
// this feature can be used in a Conditional Corpus Pipeline.
doc.getFeatures().put("has_persons", ! persons.isEmpty());
```

What do you think this will do?

Groovy Scripting PR

- Run the pipeline and note the output in the Messages tab.
- Open a few documents, examine the document features, and compare them with the annotations in the default AS.

Benchmarking

“We didn’t underperform. You overexpected.”

Why Benchmark?

- GATE provides a variety of different evaluation tools, which let you see how good your results are
- These let you compare your results against a gold standard, or compare two different annotation sets (e.g. from two different manual annotators)
- It can also be useful to compare two different versions of a system against a gold standard, to see how things have changed between different versions
- Typically, you modify the grammars to improve precision, and recall lowers, or vice versa

Corpus Benchmark Tool

- Compares annotations at the corpus level
- Compares all annotation types at the same time, i.e. gives an overall score, as well as a score for each annotation type
- Enables regression testing, i.e. comparison of 2 different versions against gold standard
- Visual display, can be exported to HTML
- Granularity of results: user can decide how much information to display
- Results in terms of Precision, Recall, F-measure

Corpus structure

- Corpus benchmark tool requires a particular directory structure
- Each corpus must have a **clean** and **marked** sub-directory
- **Clean** holds the unannotated version, while **marked** holds the marked (gold standard) ones
- There may also be a **processed** subdirectory – this is a datastore (unlike the other two)
- Generate this automatically using the tool unless you **really** know what you're doing
- Corresponding files in each subdirectory must have the same name
- You can copy the files in the marked directory to the clean one to ensure they're identical: it will ignore the marked annotations in the clean version anyway

How it works

- Clean, marked, and processed directories
- Corpus_tool.properties – must be in the directory where you run GATE from (normally top-level)
- Specifies configuration information about
  - What annotation types are to be evaluated
  - Threshold below which to print out debug info (need verbose mode set for this to function)
  - Input set name and key set name
- Modes
  - Store results for later use
  - Human marked against previously stored (processed)
  - Human marked against current processing results (current)
  - Compare both versions against marked (default mode)

Corpus Benchmark Tool

View Attachment: temp.html

ABC19980430.1830.0858.sgm

Annotation Type	Precision	Recall	Annotations
Annotation type: Organization	1.0 Precision increase on human-marked from 0.75 to 1.0	0.75 Recall increase on human-marked from 0.375 to 0.75	MISSING ANNOTATIONS in the automatic texts: **ABC**: *[2849,2852]* SPURIOUS ANNOTATIONS in the automatic texts: PARTIALLY CORRECT ANNOTATIONS in the automatic texts:
Annotation type: Person	0.9444444444444444 Precision increase on human-marked from 0.8947368421052632 to 0.9444444444444444	0.9444444444444444	
Annotation type: GPE	1.0	1.0 Recall increase on human-marked from 0.8571428571428571 to 1.0	

Analysing the Results

- Details of errors (ABC not recognised as Organization)

Annotation Type	Precision	Recall	Annotations
Annotation type: Organization	1.0 Precision increase on human-marked from 0.75 to 1.0	0.75 Recall increase on human-marked from 0.375 to 0.75	MISSING ANNOTATIONS in the automatic texts: **ABC**: *[2849,2852]* SPURIOUS ANNOTATIONS in the automatic texts: PARTIALLY CORRECT ANNOTATIONS in the automatic texts:

- Improved precision and recall since previous version

Corpus benchmark tool demo

- Setting the properties file
- Running the tool in different modes
- Visualising the results

Try it yourself if you're feeling brave!

- All files are in module-10-advanced-ie/hands-on/corpus-benchmark
- Copy corpus_tool.properties to where you run GATE from
- Tools -> Corpus Benchmark to run the tool
- *Store corpus for future evaluation*: use ANNIE (gate/plugins/ANNIE/ANNIE-with-defaults.gapp) on your selected corpus
- *Marked vs stored*: use test-corpus
- *Marked vs current*: use ANNIE-no-OM.gapp on test-corpus
- *Default*: select Verbose mode (checkbox) and use ANNIE-no-OM.gapp on test-corpus

Putting it all together

- You can combine ideas from all these topics (and more) when creating your applications
- Here's a real example of an IE application we created for a project on business intelligence
- It involved different kinds of HTML texts, which required different processing methods
- As you will see, it's important to keep calm and do things one step at a time
- When you have complex applications like this, keeping things in separate annotation sets (and removing unwanted annotations) can help you keep track of what's going on

Complex IE application

Pre-process all documents

Add document features depending on text type

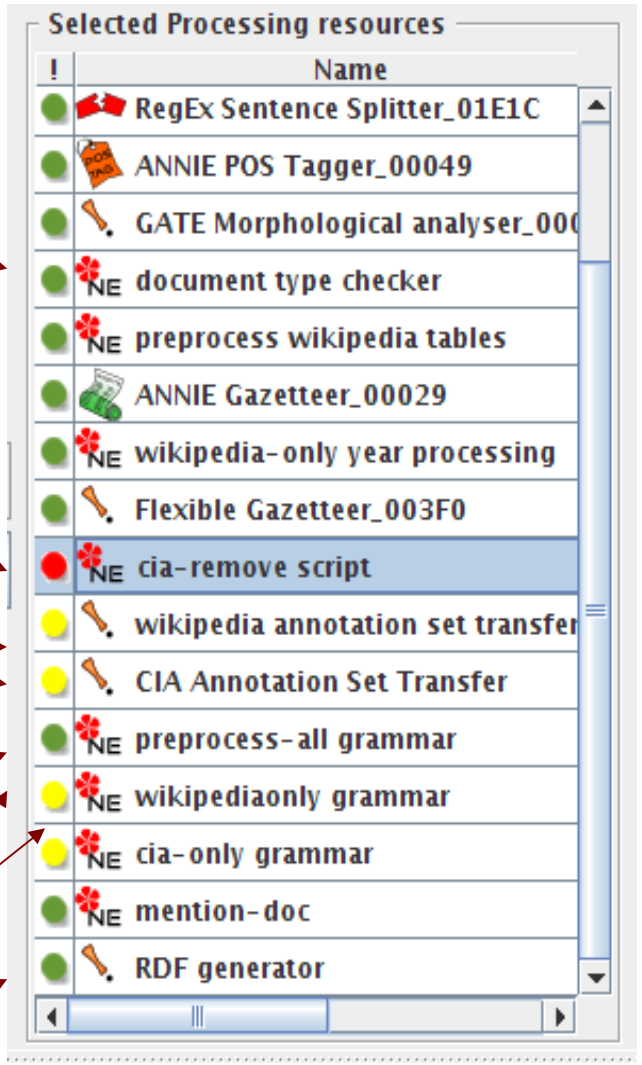
Tried this grammar out, but didn't use it ultimately

For each text type, copy the pre-processing annotations from the relevant section to a new annotation set

Pre-processing is same for all document types

Run a text-specific grammar on the documents

Do something with the results of all documents



Summary of this module

- You should now have some ideas about how to take ANNIE a step further and do more interesting things in GATE than just IE on English news texts.
- Porting an IE system to a different language
- Processing multilingual corpora
- How to process “difficult” texts, e.g. keeping sections independent, processing only parts of a document, processing large documents.
- How to manipulate existing annotated documents
- This should enable you now to start building up more complex applications with confidence

Take-home message for today

- Don't be afraid to try weird and wonderful things in GATE!
- We do it all the time...sometimes it even works :-)

Extra hands-on

- Modify ANNIE for a language of your choice, by adapting some gazetteer lists and adding some grammar rules
- If this isn't feasible for your language (e.g. Chinese) then just make an application with some simple gazetteer lists for your language and some rules which convert the Lookups into annotations
- Create a small corpus containing a combination of documents in your chosen language and in English
- Create an application that processes the documents separately but which merges the results from both into a single final annotation set
- Use any method you like to only annotate certain parts of those documents