Module 4: Machine Learning

Module 4 Outline

10-11.30am	•What is machine learning and why do we want to do it? •Setting up a corpus •Setting up a configuration file •Running the ML PR in evaluation mode
11.30-11.45	BREAK
11.45-13-15	•Evaluation in ML •Running the ML PR in training mode •Running the ML PR in application mode •Varying the configuration file
13.15-14.15	LUNCH
14.15-15.30	•Examining the model •Tuning the configuration file •Learning relations—demonstration
15.30-15.45	BREAK
15.45-16.45	TALK

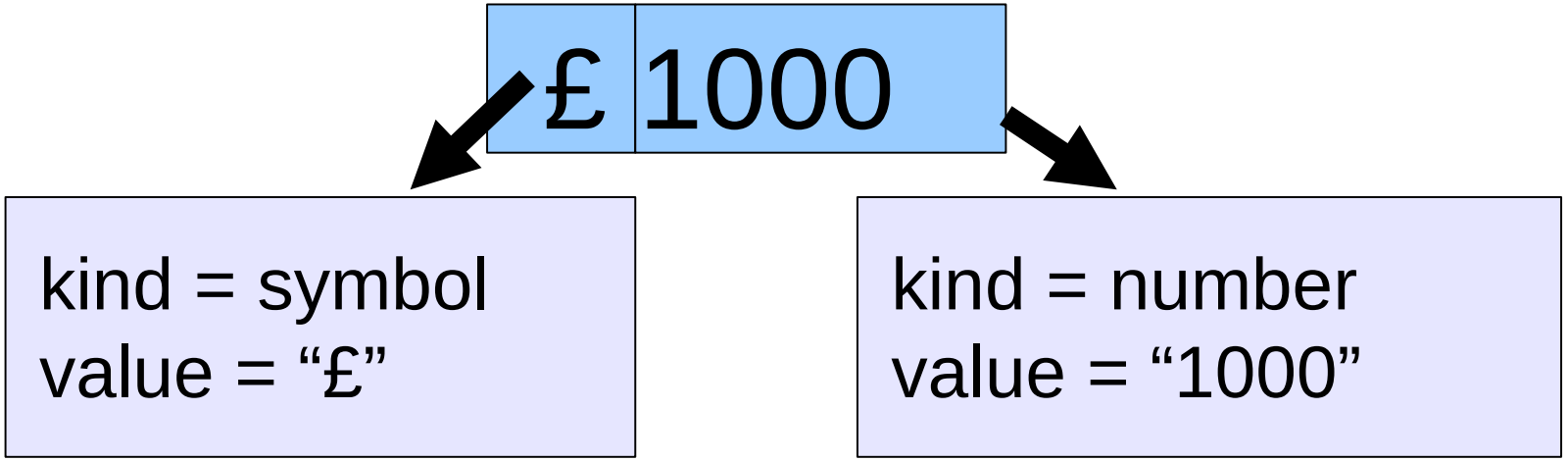
What is Machine Learning and why do we want to do it?

What is ML?

- Aim to automate the process of inferring new data from existing data
- In GATE, that means creating annotations by learning how they relate to other annotations

Learning a pattern

- For example, we have “Token” annotations with “kind” and “value” features



- ML could learn that a “£” followed by a number is an amount of currency

How is that better than making rules (e.g. JAPE)?

- It is different to the rule-based approach
- Some things humans are better at writing rules for, and some things ML algorithms are better at finding
- With ML you don't have to create all the rules
- However you do have to manually annotate a training corpus
- Rule-based approaches (e.g. JAPE) and ML work well together
  - e.g. JAPE often used extensively to prepare data for ML

Terminology: Instances, attributes, classes

California Governor Arnold Schwarzenegger proposes deep cuts.

Instances: Any annotation
Tokens are often convenient

Token | Token | Token | Token | Token | Tok | Tok

Attributes: Any annotation feature relative to instances
Token.String
Token.category (POS)
Sentence.length

Sentence

Class: The thing we want to learn
A feature on an annotation

Entity.type =Location

Entity.type=Person

Instances

- Instances are cases that may be learned
- Every instance is a decision for the ML algorithm to make
- To which class does this instance belong?

Attributes

- Attributes are pieces of information about instances
- They are sometimes called “features” in machine learning literature

Classes

- The class is what we want to learn
- For example, if we want to find person names, for every instance, the question is, is this a person name?
  - The classes are “yes” and “no”
- Sometimes there are many classes, for example we may want to learn entity types
  - For every instance, the question is, which of a predetermined entity type set does this belong to?

Training

- Training involves presenting data to the ML algorithm from which it creates a model
- The training data (instances) have been annotated with class annotations as well as attributes
- Models are representations of decision-making processes that allow the machine learner to decide what class the instance has based on the attributes of the instance
- Models are covered in more detail in Module 11 (Advanced Machine Learning)

Application

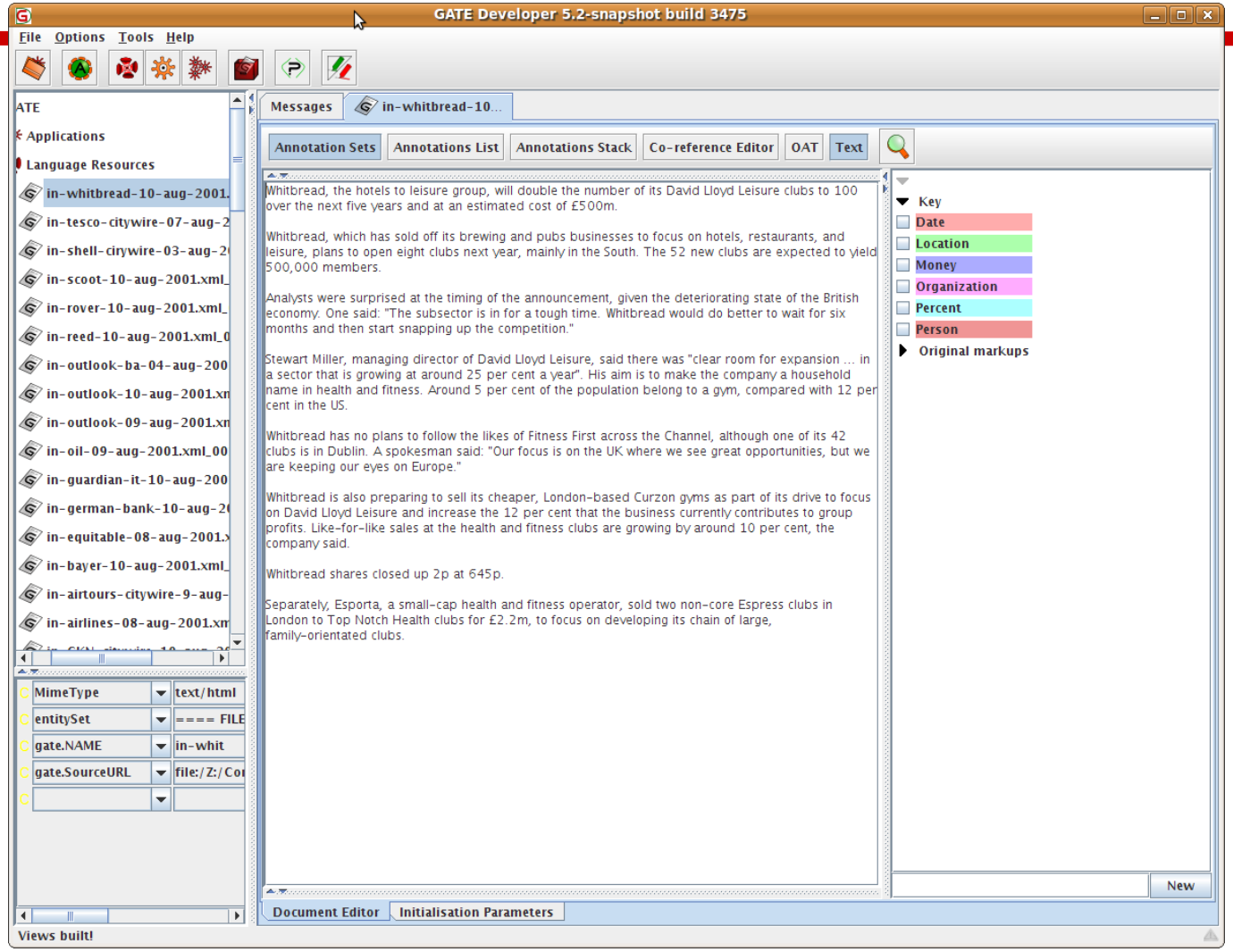
- When the machine learner is applied, it creates new class annotations on data using the model
- The corpus it is applied to must contain the required attribute annotations
- The machine learner will work best if the application data is similar to the training data

Evaluation

- We want to know how good our machine learner is before we use it for a real task
- Therefore we apply it to some data for which we already have class annotations
  - The “right answers”, sometimes called “gold standard”
- If the machine learner creates the same annotations as the gold standard, then we know it is performing well
- The test corpus must not be the same corpus as you trained on
  - This would give the machine learner an advantage, and would give a false idea of how good it is

Setting up a Corpus

Load the Corpus



- **Create a corpus and populate from your hands-on materials**
- **Use UTF-8 encoding**
- **Open a document and look at the annotations it contains**
- It doesn't matter what you call the corpus

Examining the corpus

- The corpus contains an annotation set called “Key”
- Within this annotation set are annotations of types “Date”, “Location”, “Money”, “Organization” and so forth
- There are also some original markups
- The annotation set “Key” has been manually prepared with some entity types

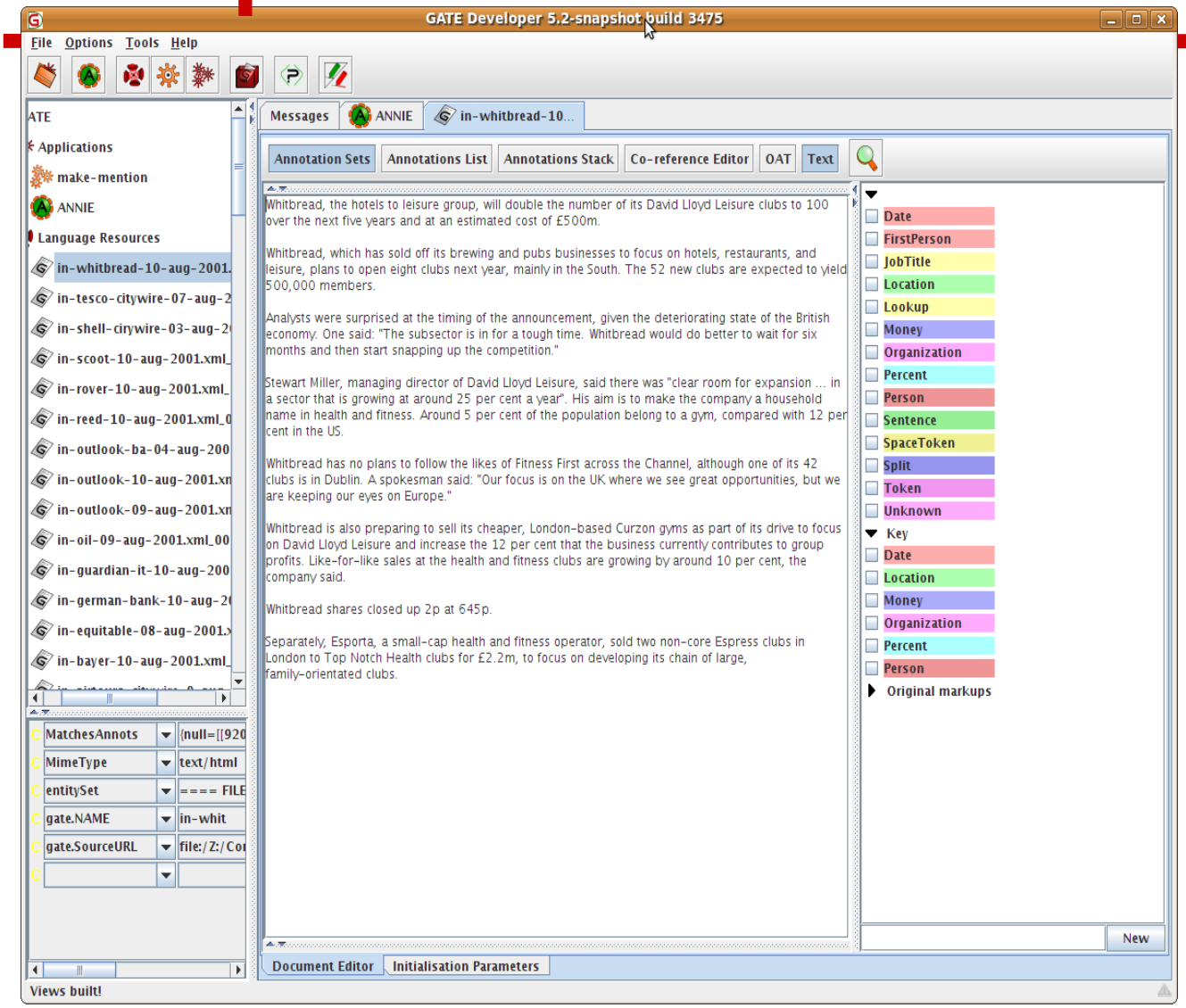
What are we going to use this corpus for?

- We are going to train a machine learner to annotate corpora with these entity types
- We need a training corpus and a test corpus
- The training corpus will be used by the machine learner to deduce relationships between attributes and entity types (classes)
- The test corpus will be used to find out how well it is working, by comparing annotations created by the learner with the class annotations that are already there

Instances and Attributes

- This corpus so far contains only the class annotations
- There is not much in this corpus to learn from
- What would our instances be?
- What would our attributes be?
- If we run ANNIE over the corpus, then we can use “Token” annotations for instances, and we would have various options for attributes
- **Run ANNIE over your corpus**
- **Exclude the Key annotation set from the document reset PR!**

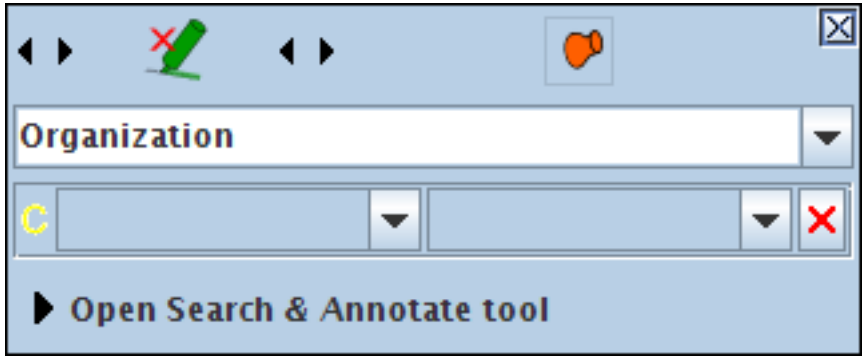
Running ANNIE on the corpus



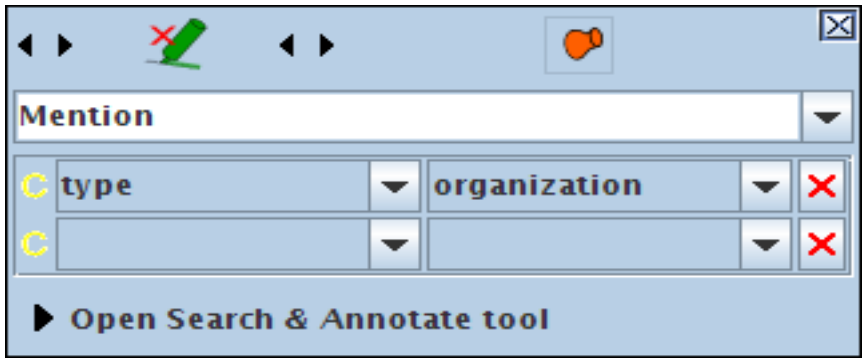
- Having run ANNIE on the corpus, we have more annotations to work with

Preparing the corpus: Classes

- What we have:



- What we need:



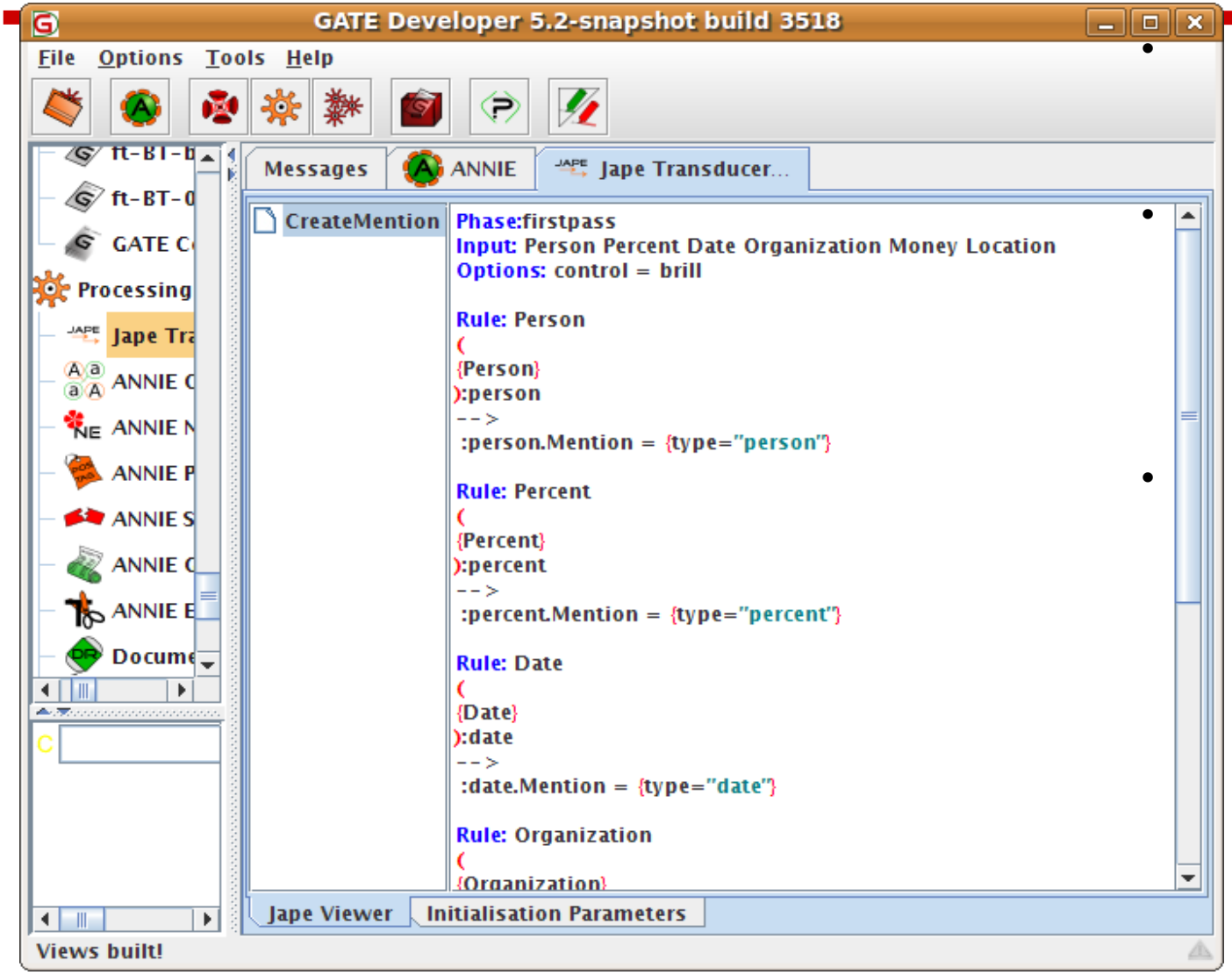
Preparing the corpus: Classes

- Currently each class has its own annotation type (Date, Person, Percent etc.)
- However ML PR expects class to be a feature value, not a type
- Therefore we are going to make a new annotation type for the ML to learn from, e.g. “Mention”, though it does not matter what it is called

Making class annotations

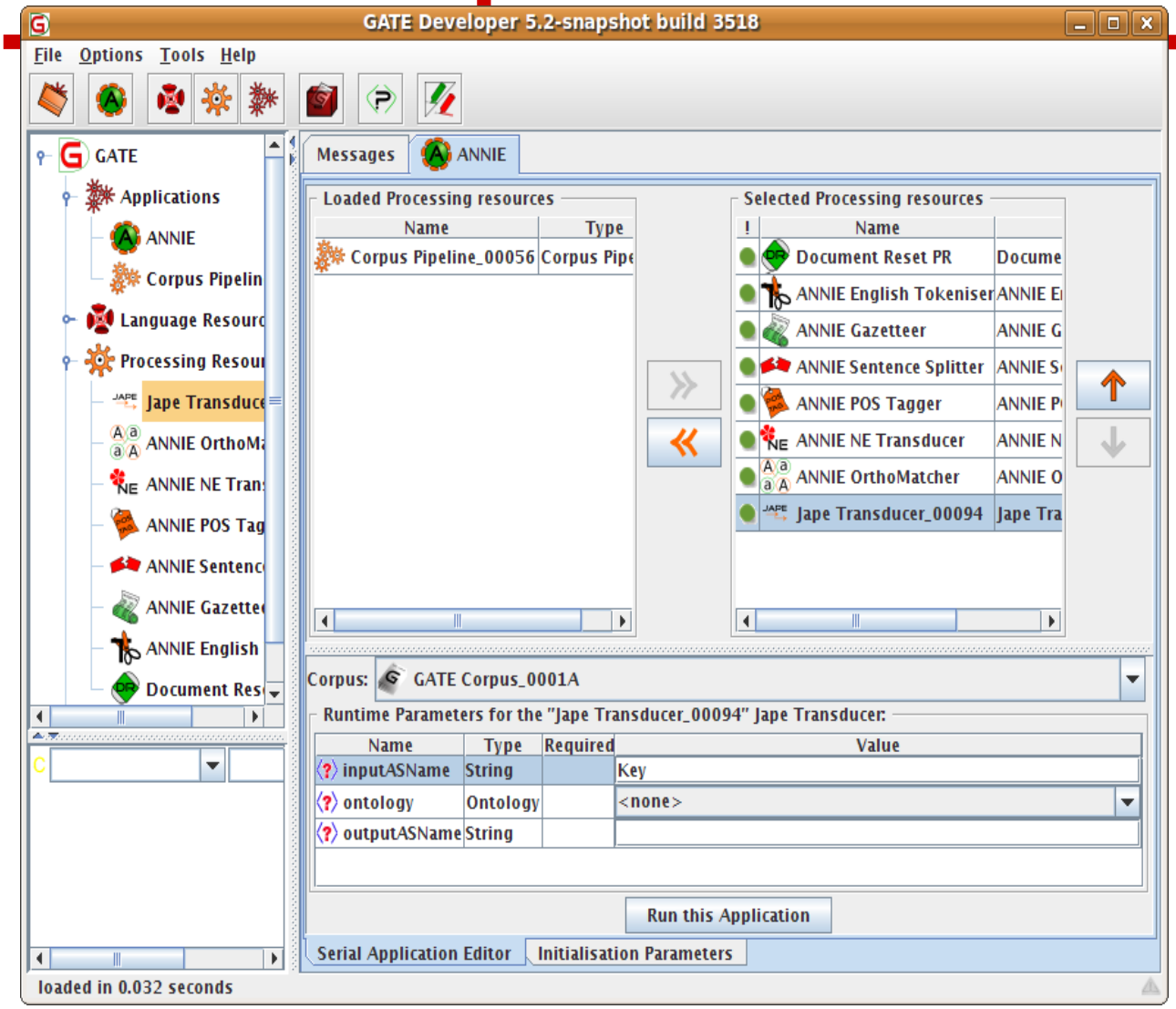
- Load a JAPE transducer with the “CreateMention.jape” grammar that you will find in your hands-on materials
- Look at the grammar

The CreateMention.jape grammar



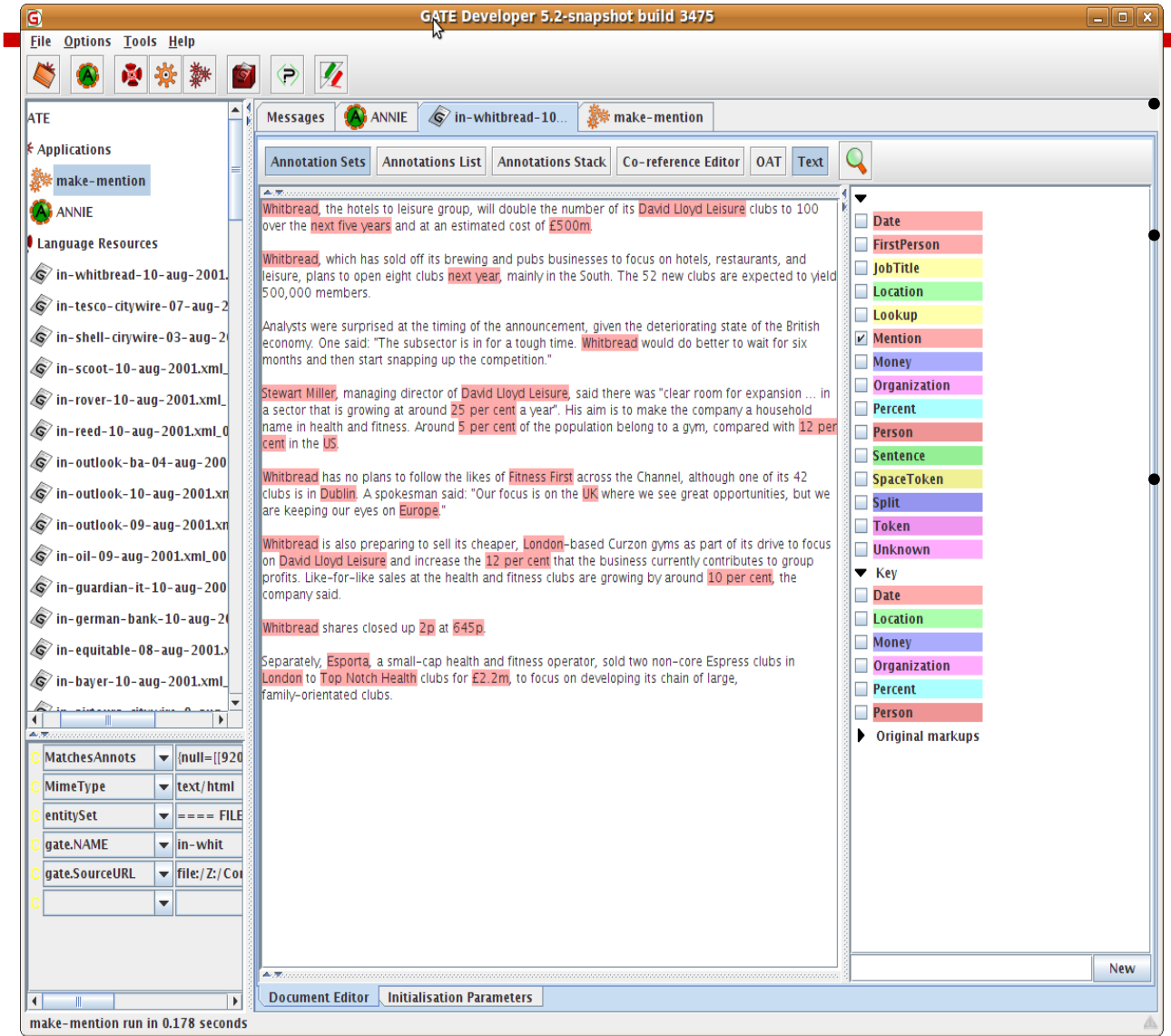
- This grammar makes a new annotation type called “Mention”
- It makes the previous annotation type into a feature of the “Mention” annotation
- Feature name is “type” because “class” is reserved for ontologies

Applying the grammar to the corpus



- Add the JAPE transducer at the end of your ANNIE application
- Set the inputASName to “Key”
- Leave the outputASNam e blank

Check the “Mention” annotations



Rerun the application

Check that you have some “Mention” annotations

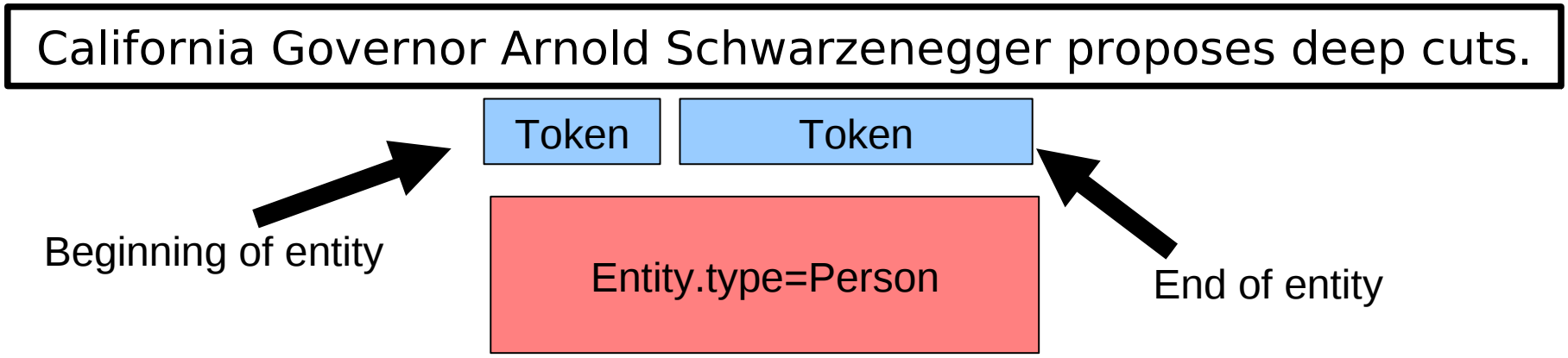
Check that they have a feature “type” and that the values look right

The Configuration File

Looking at the configuration file

- In the configuration file, we tell the machine learning PR what we want it to do
- You will find a configuration file in your hands-on materials, called ml-config-file.xml
- **Open it using a text editor**

<SURROUND value="true"/>



- This learned class covers more than one instance....
- Begin / End boundary learning
- Dealt with by API - ***surround mode***
- Transparent to the user

Confidence Thresholds

```
<PARAMETER name="thresholdProbabilityEntity" value="0.2"/>
<PARAMETER name="thresholdProbabilityBoundary" value="0.4"/>
```

- Learner will provide confidence ratings—how likely is a result to be correct
- We must determine how certain is good enough
- Depending on the application we might prefer to include or exclude annotations for which the learner is not too sure
- thresholdProbabilityBoundary is a threshold for the beginning and end instances
- thresholdProbabilityEntity is a threshold for beginning and end instances combined

<multiClassification2Binary method="one-vs-others"/>

California Governor Arnold Schwarzenegger proposes deep cuts.

Entity.type =Location

Entity.type=Person

- Many algorithms are binary classifiers (e.g. yes/no)
- We have several classes (Person, Location, Organization etc.)
- Therefore the problem must be converted so that we can use binary algorithms to solve it
- **one-vs-others**
  - LOC vs PERS+ORG / PERS vs LOC+ORG / ORG vs LOC+PERS
- **one-vs-another**
  - LOC vs PERS / LOC vs ORG / PERS vs ORG

<EVALUATION method="holdout" ratio="0.66"/>

- We are going to evaluate our application in two ways today
- The ML PR will evaluate for us
- We will also run our own evaluation
- This parameter dictates how the ML PR will evaluate for us, if we run it in evaluation mode
- We are telling it that it should reserve a third of the data as a test set, train, then apply the result to the held out set
- Alternatively, we could ask the PR to run a cross-validation evaluation

K-Fold Cross-Validation

- In k-fold cross-validation, the corpus is split into k equal parts
- Each part is held back as test data and the learner trained on the remainder
- The learner is then tested on the held-out portion
- This is repeated for each part
- This maximises the training data without losing testing accuracy
- Example:

```
<EVALUATION method="kfold" runs="10"/>
```

<ENGINE nickname="PAUM" ..

- Next we specify what machine learning algorithm we wish to use
- Today we are using the perceptron with uneven margins (“PAUM”)
- We will use the following options: options="-p 50 -n 5 -optB 0.3"
  - Challenge: find out what these options do!

<INSTANCE-TYPE>Token</INSTANCE-TYPE>

- Next, we tell the ML PR what our instance annotation is
- The goal of the ML PR is, for every instance, to try to learn how the attributes of the instance relate to its class
- So the instance is a very critical concept
- We have decided that the “Token” is our instance annotation type
  - We made sure, earlier, that we have “Token” annotations in our corpus

Specifying Attributes

```
<ATTRIBUTELIST>
<NAME>Form</NAME>
<SEMTYPE>NOMINAL</SEMTYPE>
<TYPE>Token</TYPE>
<FEATURE>category</FEATURE>
<RANGE from="-2" to="2"/>
</ATTRIBUTELIST>
```

- For every attribute, we create a specification like the one above
- This is the information from which the PR will learn, so it is important to give it some good data
- You can see in the configuration file that there are several attributes, providing a good range of information
- However, if you have too many attributes it can take a very long time to learn!

Breaking down the attribute specification

- <NAME>Form</NAME>
  - This is the name that we choose for this attribute. It can be anything we want, but it will help us later if we make it something sensible!
- <SEMTYPE>NOMINAL</SEMTYPE>
  - Is the value of this attribute a number or a name?

Breaking down the attribute specification

- <TYPE>Token</TYPE>
  - The value of the attribute will be taken from the “Token” annotation
- <FEATURE>category</FEATURE>
  - The value of the attribute will be taken from the “category” feature

Breaking down the attribute specification


```
<ATTRIBUTELIST>
  :
  <RANGE from="-2" to="2"/>
</ATTRIBUTELIST>
```

- Because this is an “ATTRIBUTELIST” specification, we can specify a “RANGE”
- In this case, we will gather attributes from the current instance and also the preceding and ensuing two

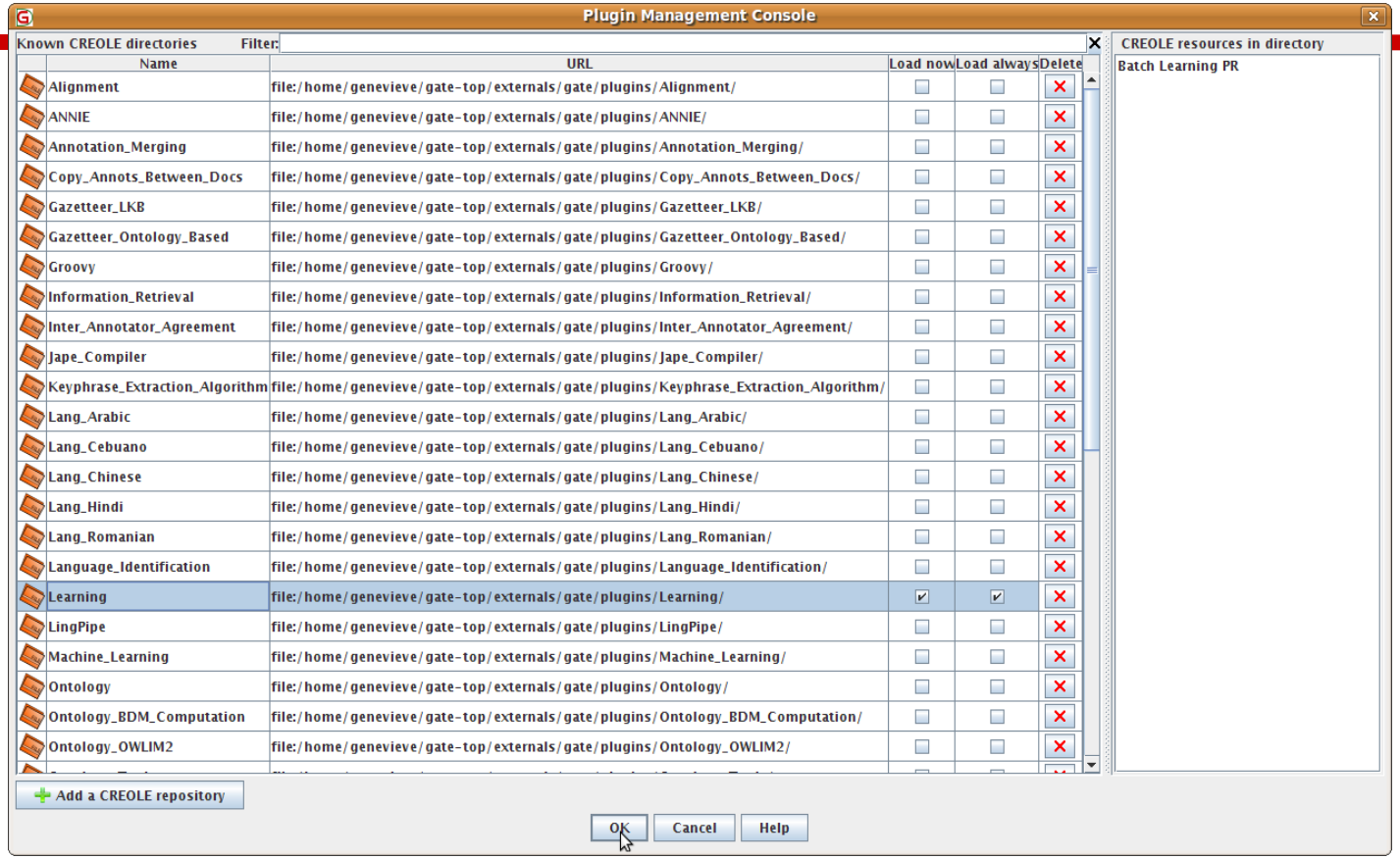
Specifying the Class Attribute


```
<ATTRIBUTE>
    <NAME>Class</NAME>
    <SEMTYPE>NOMINAL</SEMTYPE>
    <TYPE>Mention</TYPE>
    <FEATURE>class</FEATURE>
    <POSITION>0</POSITION>
    <CLASS/>
</ATTRIBUTE>
```

- This attribute is the class attribute
- You can call it whatever you want, but “Class” is a sensible choice!
- Remember that our class attribute is in the “Mention” annotation type, in the “class” feature
- This is an “ATTRIBUTE”, not an “ATTRIBUTELIST”, so we have only “position”, not “range”
- Saying <CLASS/> tells the ML PR that this is the class attribute. This is what it has to learn.

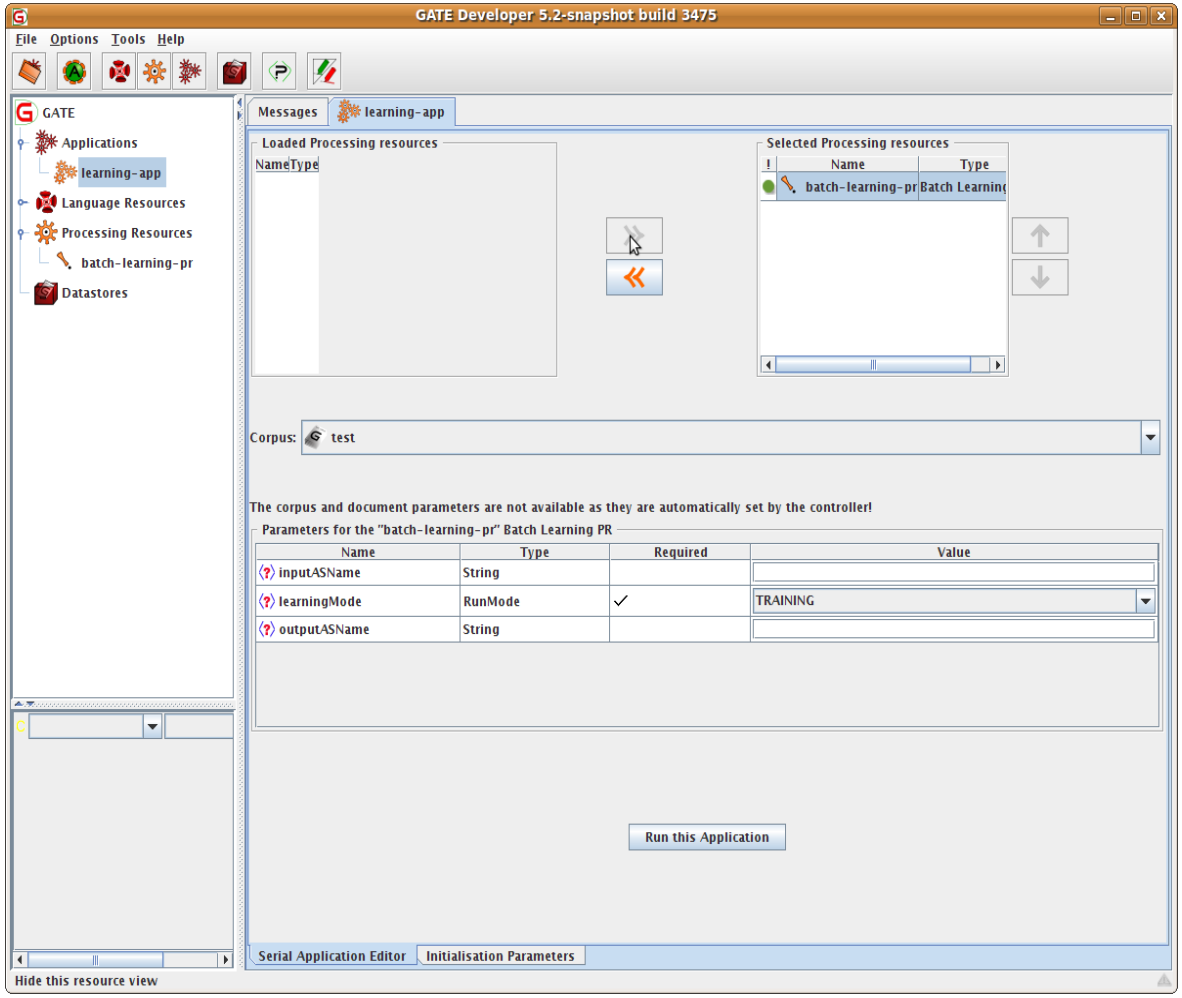
Running the ML PR in evaluation mode

Loading the Learning plugin



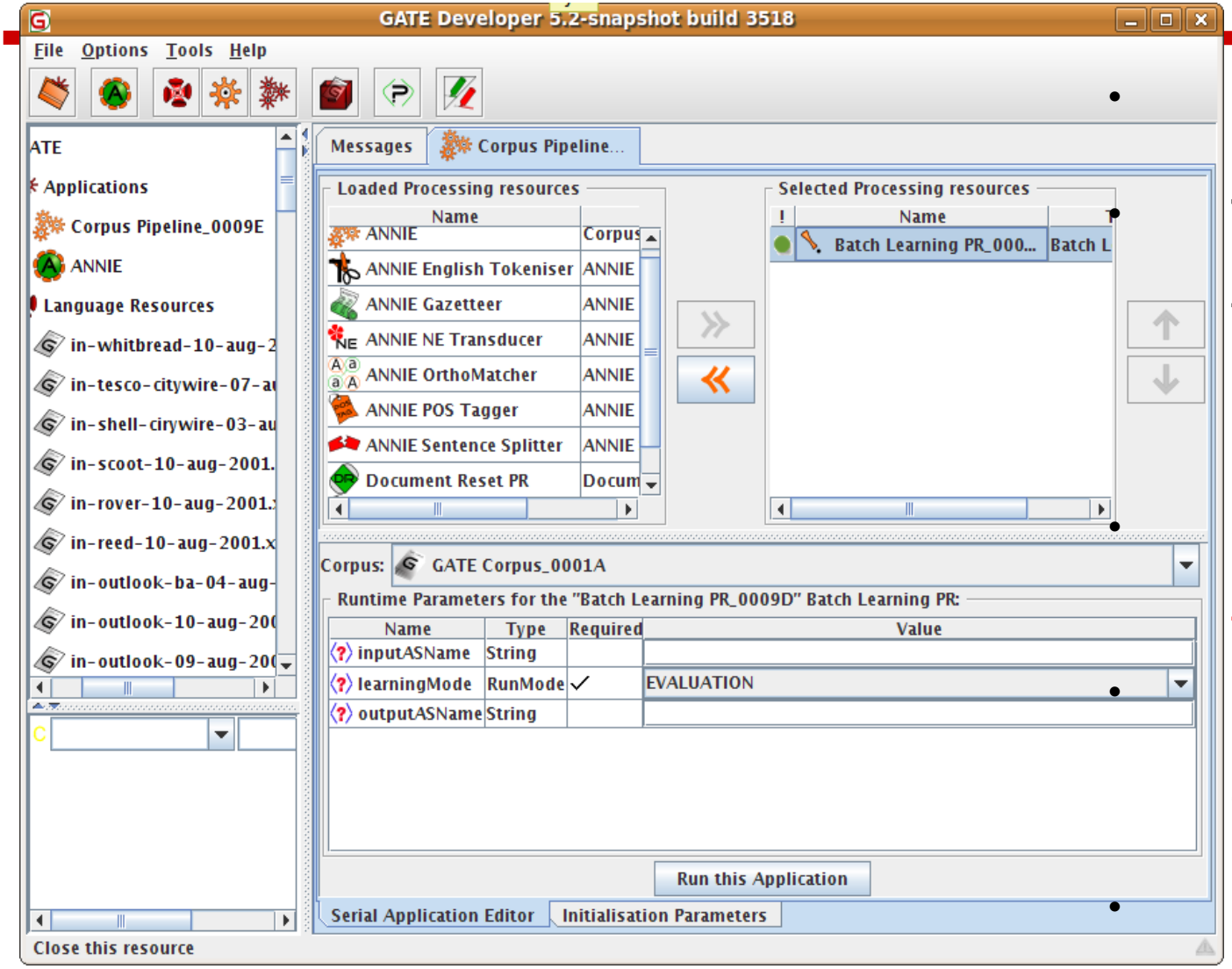
- **Load the “Learning” plugin**
- (We are **not** going to work with the “Machine Learning” plugin because this is an older plugin which does not have all the functionality we want to use.)

Creating a learning application



- Create a “Batch Learning PR” using your configuration file
- Make a corpus pipeline application, and put the PR into it

Running the application in evaluation mode



Make sure the corpus is selected

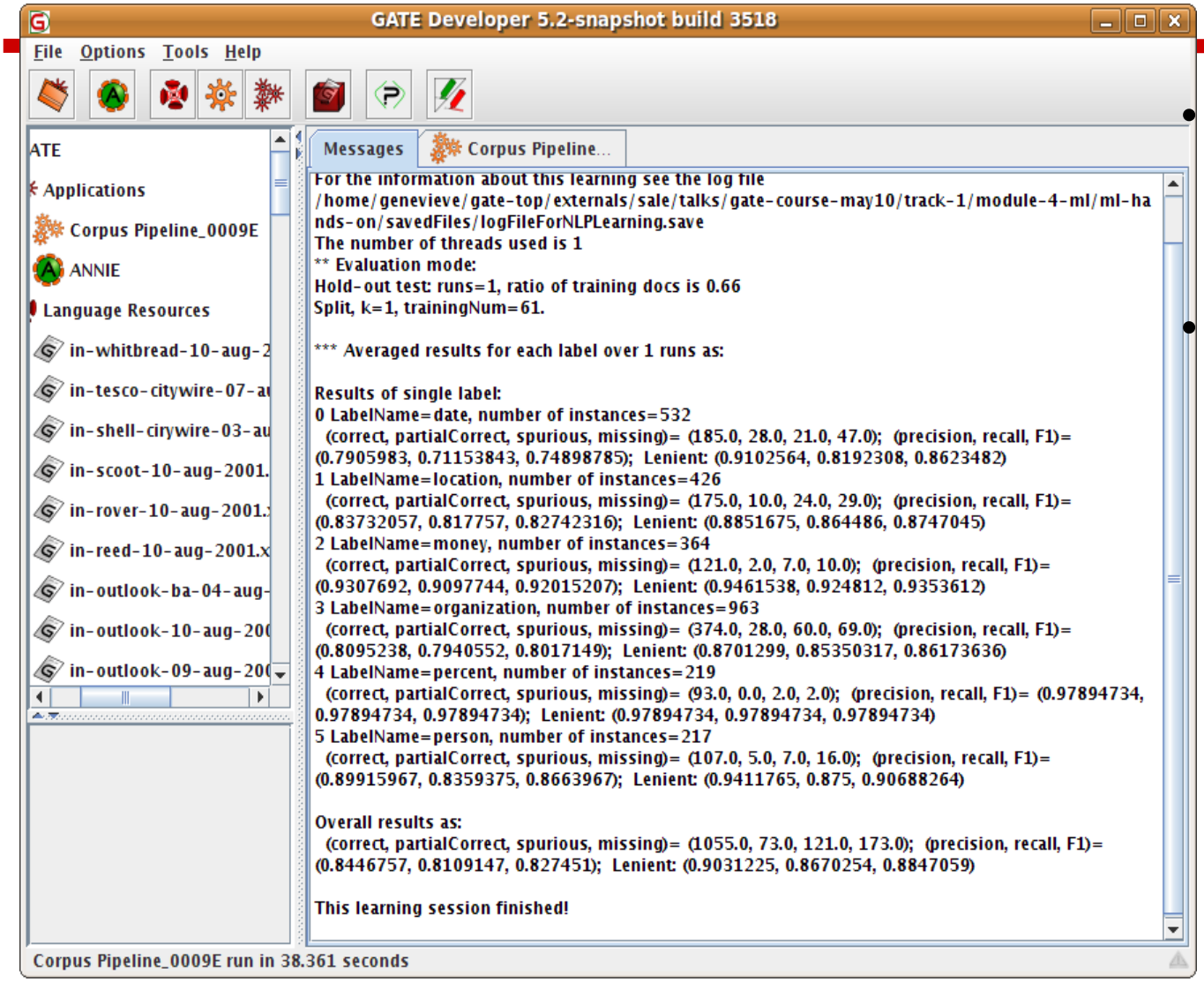
The inputASName is blank because the attributes and class are in the default annotation set

Select "EVALUATION" for the learningMode

OutputASName should be the same as inputASName in evaluation mode

Run the application!

Inspecting the results



The application may take a few minutes to run

When it is finished, switch to the "Messages" tab to examine the results

How well did we do?

- Here is my result:

(precision, recall, F1)= (0.8446757, 0.8109147, 0.827451)

- These figures look pretty good
- But what do they really mean?
- Next we will discuss evaluation measures
- Then we will run the PR in different modes
- Then we will see if we can get these numbers any higher!

Evaluation in Machine Learning

Recap of Evaluation in GATE

- Recall from Module 2 that evaluation is an important part of information extraction work
  - We need to find out how good our application is by comparing its annotations to the “right answers” (manually prepared annotations)
  - Sometimes we need to compare annotations by different annotators, to see how consistent they are
- The methodology is pretty similar for both these cases

EVALUATION MODE

- Before the break, we ran the machine learning PR in evaluation mode
- We specified how the PR should run evaluation in the configuration file
- Once we had run the application, we obtained evaluation statistics in the “Messages” tab

Our output

- I got the following result (yours may be slightly different due to the exact document set used)

(precision, recall, F1)= (0.8446757, 0.8109147, 0.827451)

- We have a precision, a recall and an F1 figure

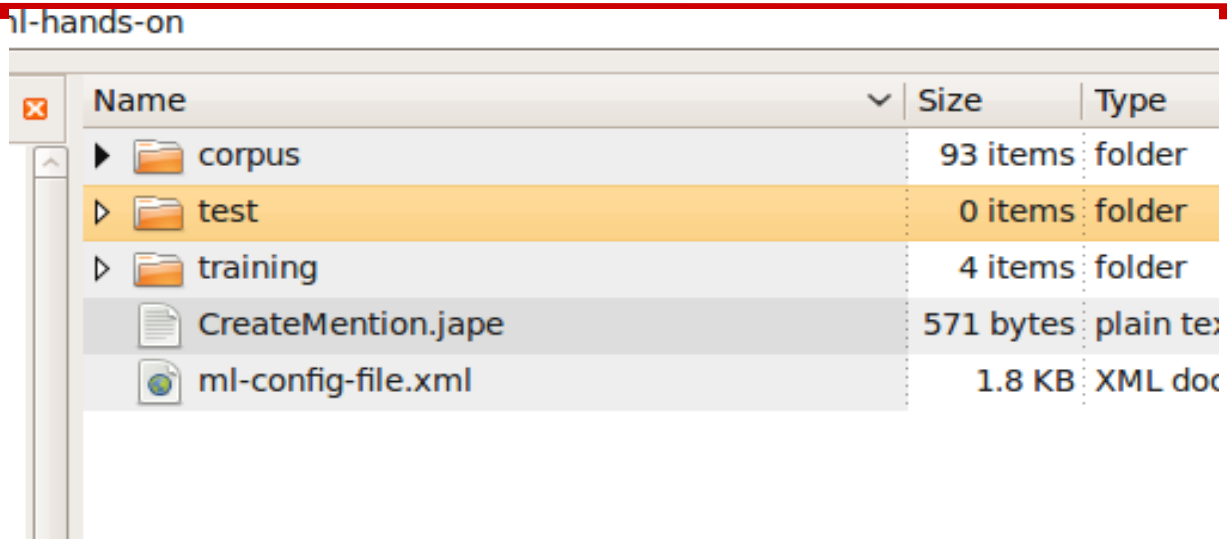
What are precision, recall and F1?

- These are the same measures we calculated using the Corpus QA tool in Module 2
- Precision is the proportion of annotations the ML PR created that were correct
- Recall is the proportion of correct annotations that the ML PR created
- F1 is an amalgam of the two measures
  - The 1 indicates that precision and recall are weighted equally
- We can equally well run our own ML evaluation using the Corpus QA tool—let's do that now

Splitting into training and test corpora

- As mentioned earlier, to truly know how well a machine learner is performing, you need to test it on data that it was not trained on
- We need separate test and training corpora
- So now we are going to split our corpus in two

Saving and splitting the corpus

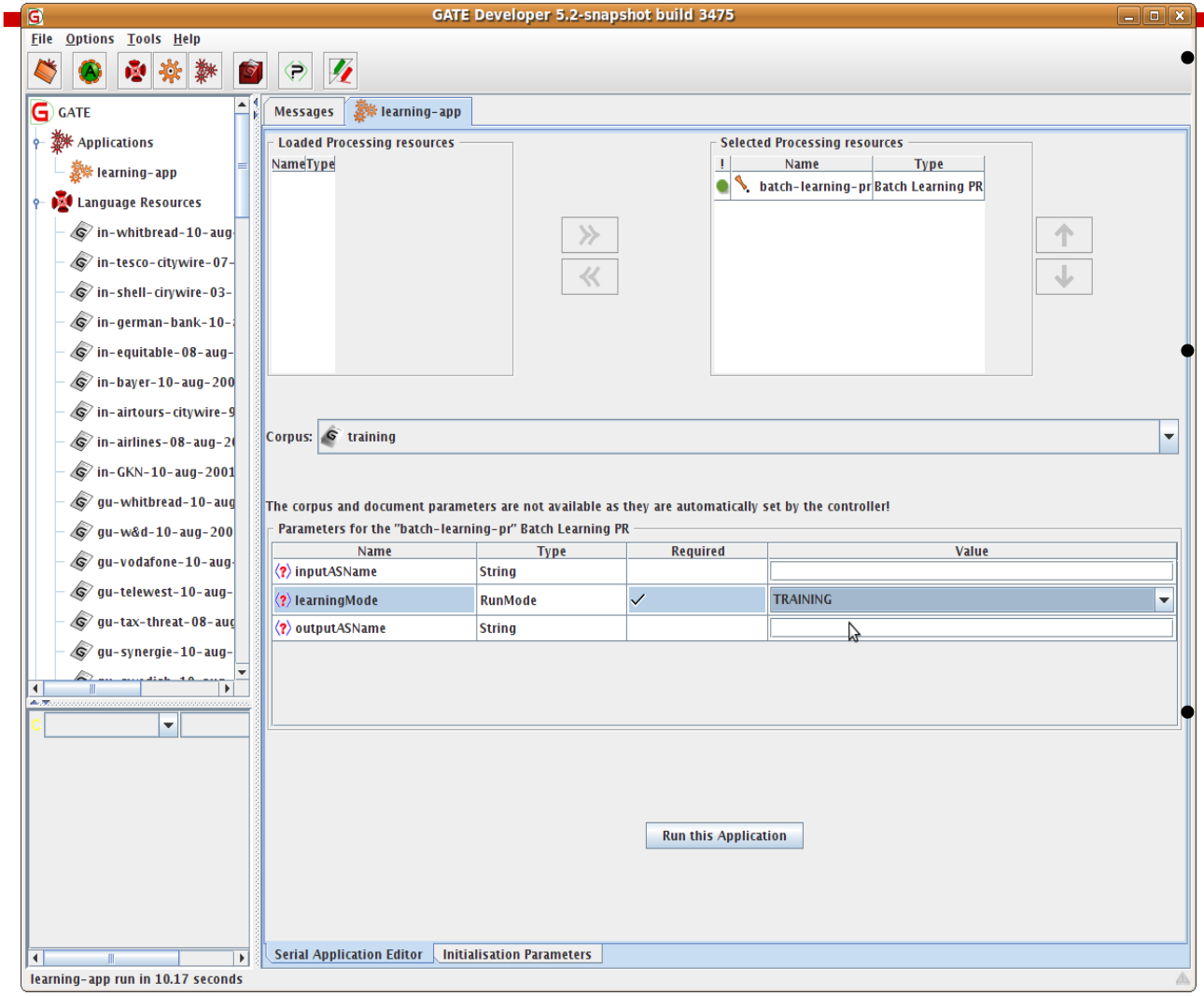


- Right click on your corpus and select “Save as XML”
- Create a new folder called “training” and save the documents into this folder
- Create a new directory alongside it called “test”
- In the file manager, cut half the documents out of “training” and paste them into “test”
  - Try to randomise them a bit!

Tidying up

- Close all your open documents and processing resources in GATE Developer
- Close your applications recursively
- Create new corpora called “training” and “test”
- Populate your corpora with the documents you saved to disk
  - As before, use UTF-8

Running the ML PR in Training Mode

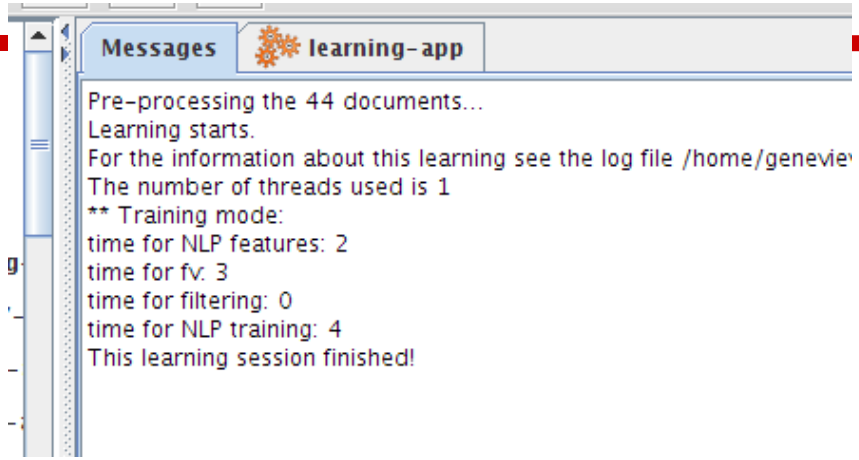


Check that your PR is set to run on the training corpus

Change the learningMode to “TRAINING” (the outputASName doesn't matter)

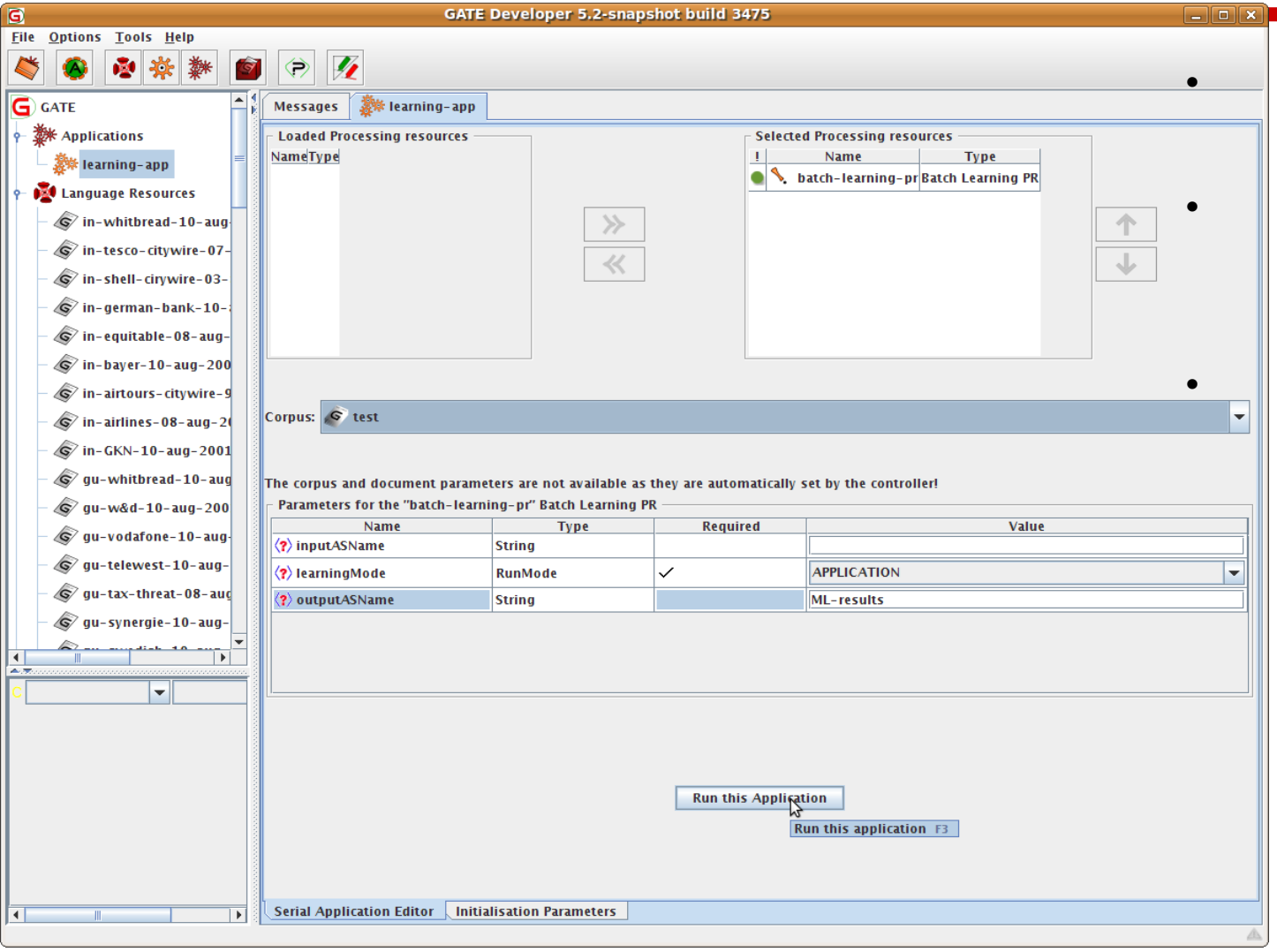
Run the application

Finished Training!



- Training may take a few minutes
- This time there is no evaluation result in the messages tab

Running the ML PR in Application Mode

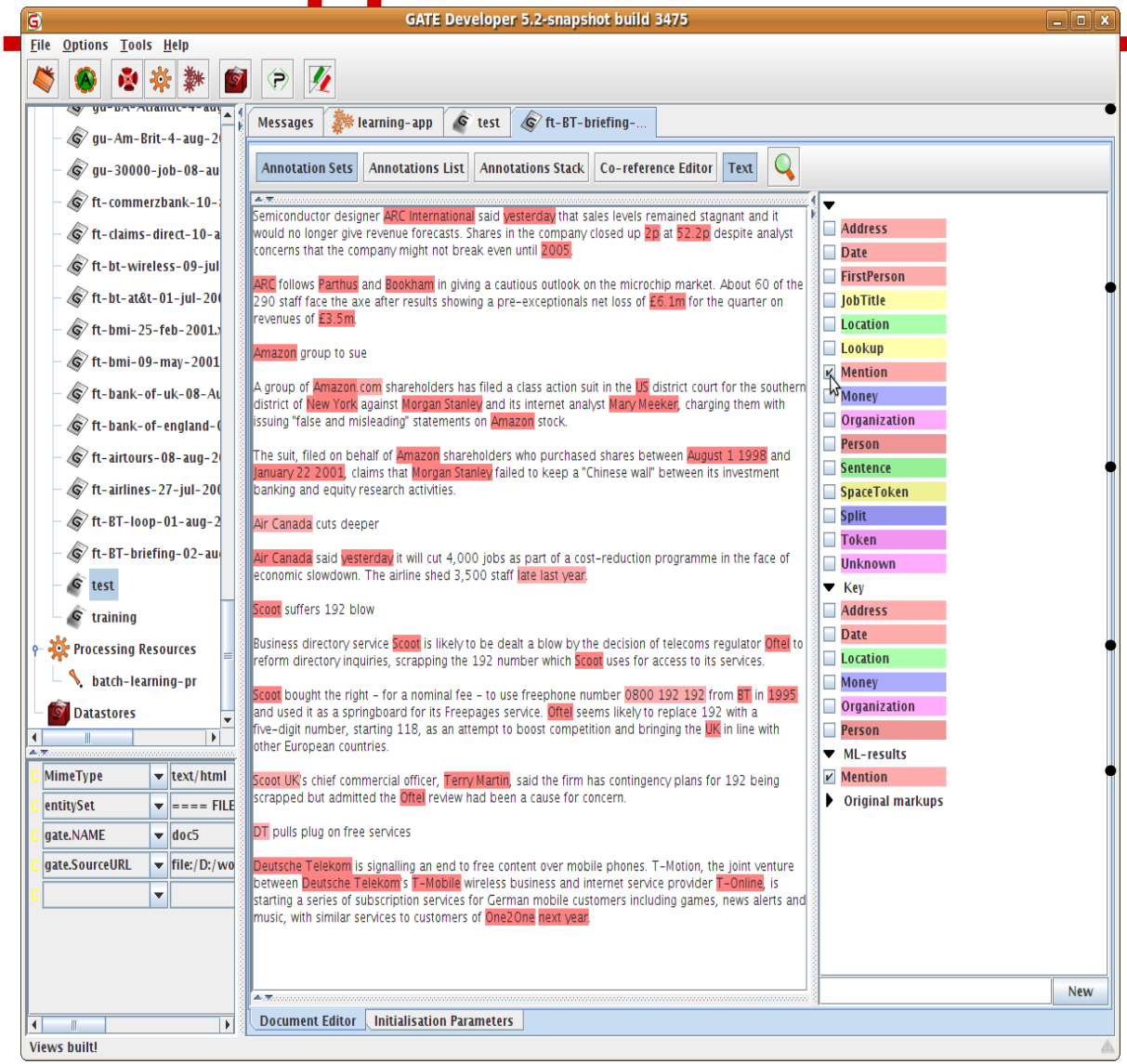


Change corpus to "test"

Change learningMode to "APPLICATION"

Set outputASName to something (e.g. "ML")--this is where your new annotations will go! You don't want to get them mixed up with the existing ones!

Examining the results of application



Choose a document from the test corpus to look at

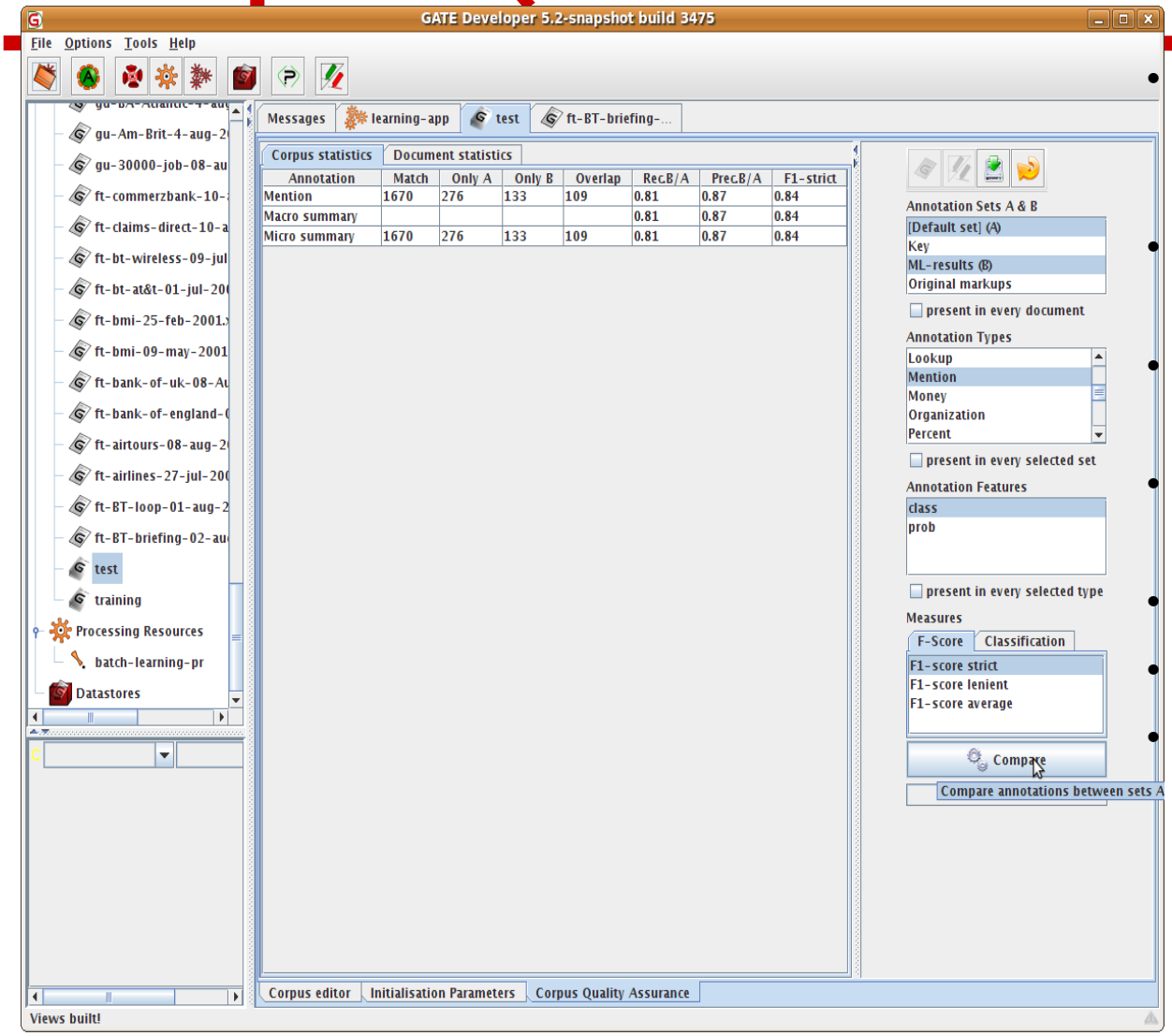
You should have a new annotation set, created by the ML application

There will be a “Mention” type both in the new set and the original

They are similar but not identical!

How similar do they appear to be? Do you think you will get a good result?

Comparing the Sets with Corpus QA



Select the test corpus and click on the Corpus QA tab

Select the Default and ML annotation sets

Select the "Mention" type

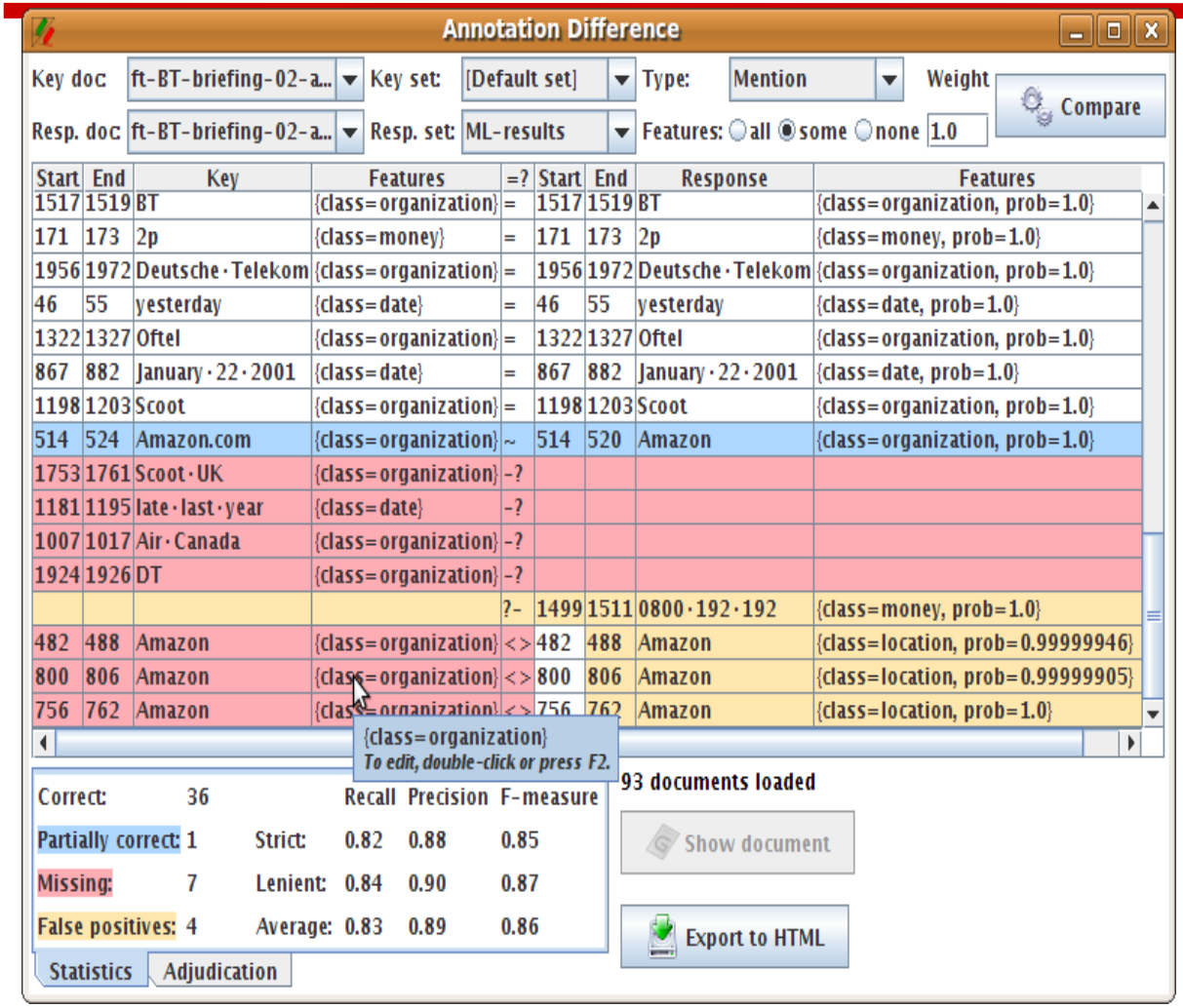
Select the "class" feature

Choose an F-measure

Click on Compare

Did you get a good result? How does it compare to the result you got using evaluation mode?

Using Annotation Diff to examine performance



- **Switch to the "Document statistics" tab**
- **Choose a document**
- **Click on the Annotation Diff icon**
- What kind of mistakes did your application make?

Varying the configuration file

- Now we are going to experiment with varying the configuration file to see if we can produce varied results
- You can edit the configuration file in your text editor
- Make sure you save your changes then **reinitialise the PR**!

Exercises

- Spend some time working on your exercise sheet
- Feel free to ask questions

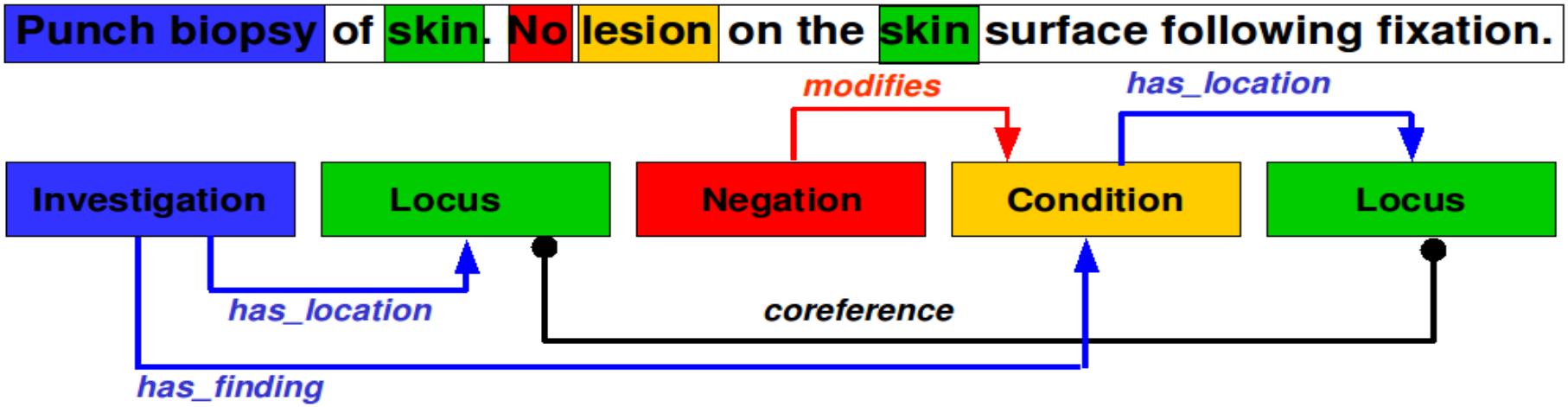
Exercises

- **Continue working on your exercise sheet until 15.00**
  - Then we will talk about learning relations

Learning relations

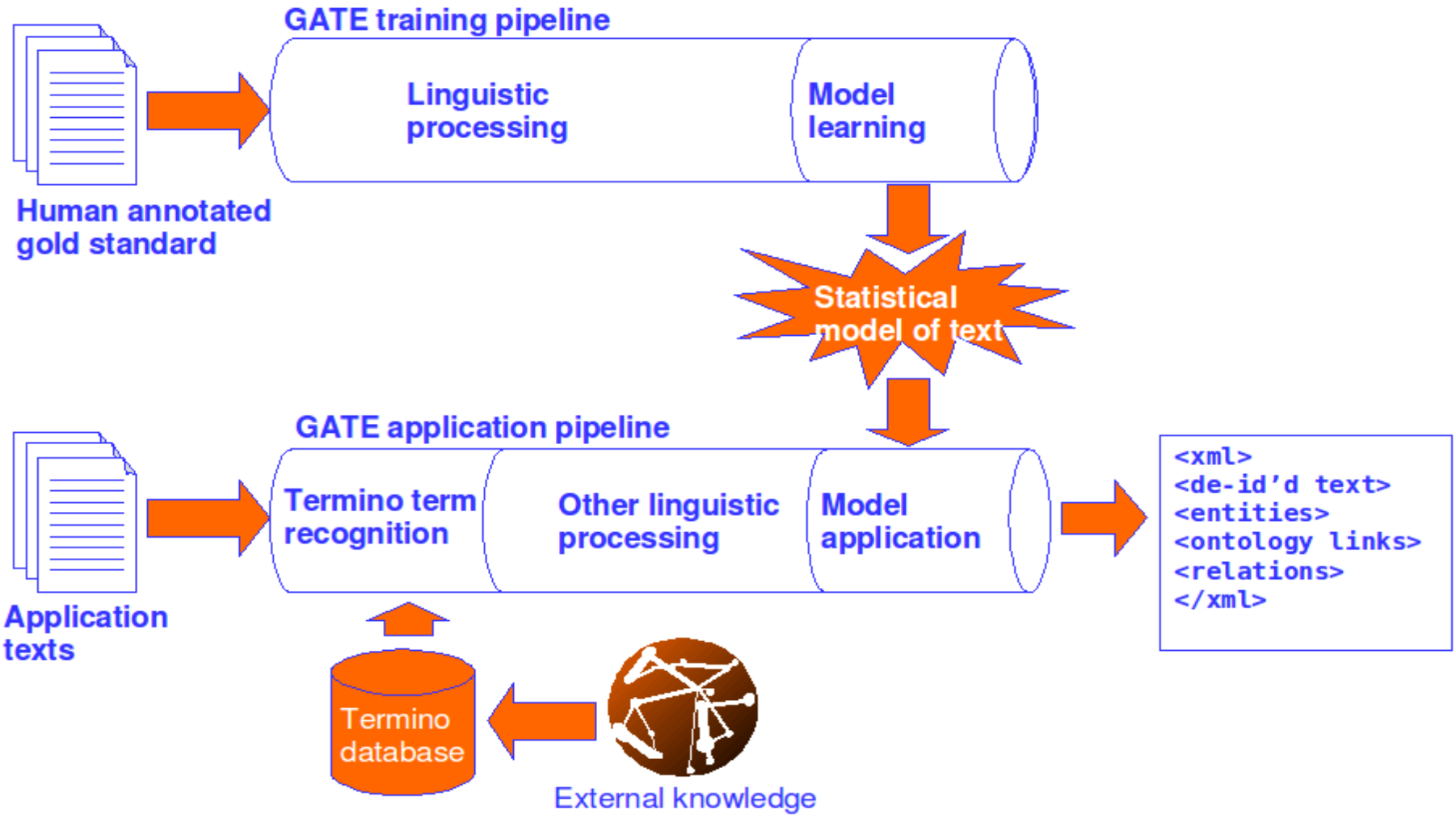
Demonstration

Entities, modifiers, relations, coreference



- The CLEF project
- More sophisticated indexing and querying
- Why was a drug given?
- What were the results of an exam?

Supervised system architecture



Previous work

- Clinical relations have usually been extracted as part of a larger clinical IE system
- Extraction has usually involved syntactic parses, domain-specific grammars and knowledge bases, often hand crafted
- In other areas of biomedicine, statistical machine learning has come to predominate
- We apply statistical techniques to clinical relations

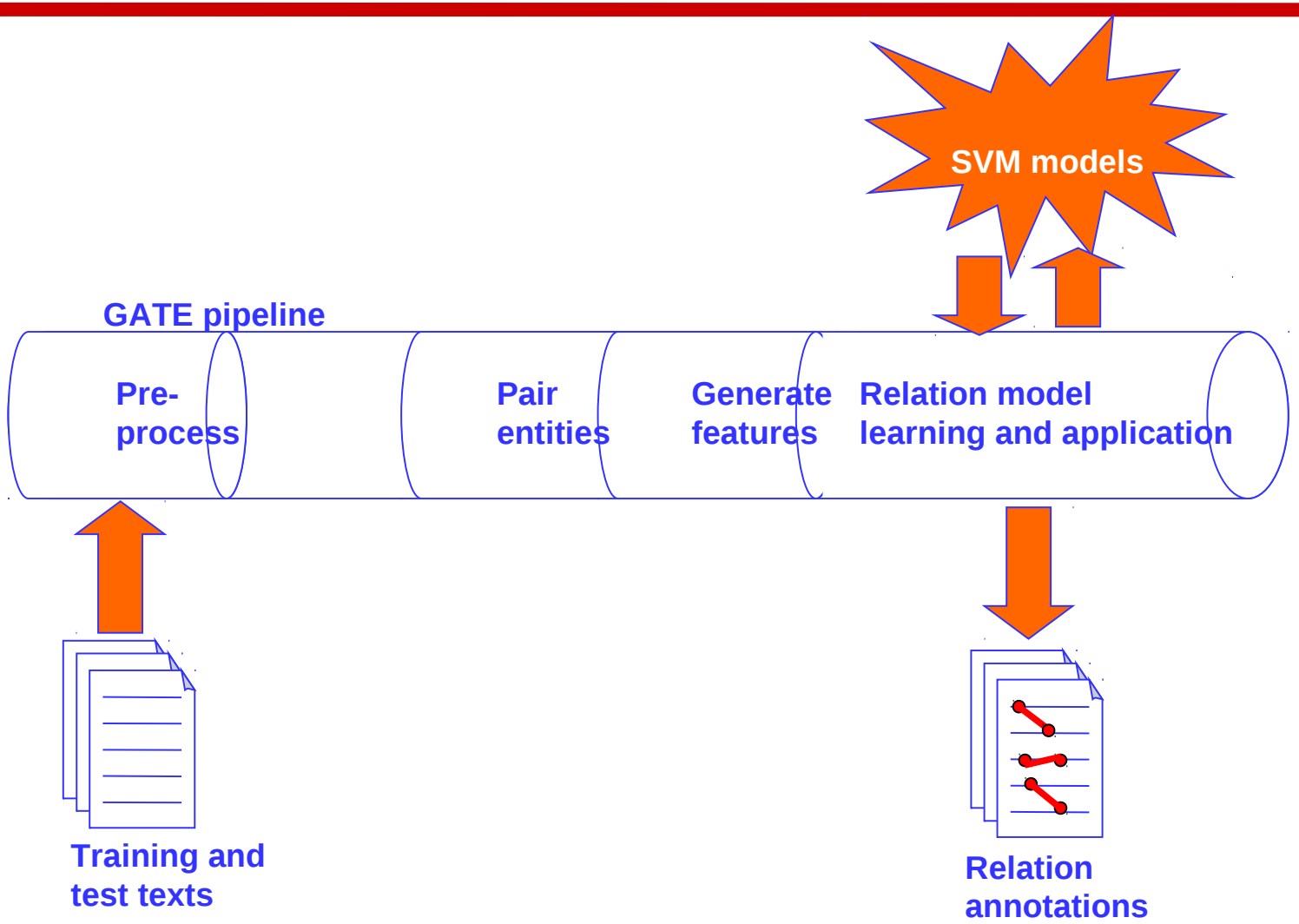
Entity types

Entity type	Brief description
Condition	Symptom, diagnosis, complication, etc.
Drug or device	Drug or some other prescribed item
Intervention	Action performed by a clinician
Investigation	Tests, measurements and studies
Locus	Anatomical location, body substance

Relation types

Relationship	Argument 1	Argument 2
has_target	Investigation	Locus
	Intervention	Locus
has_finding	Investigation	Condition
	Investigation	Result
has_indication	Drug or device	Condition
	Intervention	Condition
	Investigation	Condition
has_location	Condition	Locus
negation_modifies	Negation modifier	Condition
laterality_modifies	Laterality modifier	Intervention
	Laterality modifier	Locus
sub-location_modi	Sub-location modifier	Locus

System architecture



Learning relations

- Learn relations between pairs of entities
- Create all possible pairings of entities across *n* sentences in the gold standard, constrained by legal entity types
  - *n*: e.g. the same, or adjacent
- Generate features describing the characteristics of these pairs
- Build SVM models from these features

Configuring in GATE

```
<DATASET>
  <INSTANCE-TYPE>theInstanceAnnotation</INSTANCE-TYPE>
  <INSTANCE-ARG1>featureForIdOfArg1</INSTANCE-ARG1>
  <INSTANCE-ARG2>featureForIdOfArg2</INSTANCE-ARG2>

  <FEATURES-ARG1>...</FEATURES-ARG1>

  <FEATURES-ARG2>...</FEATURES-ARG2>

  <ATTRIBUTE_REL>...</ATTRIBUTE_REL>

  <ATTRIBUTE_REL>...</ATTRIBUTE_REL>

  ...

</DATASET>
```

Creating entity pairings

- Entity pairings provide instances
- They will therefore provide features
- A “pairing and features” PR or JAPE needs to be run before the Learning
- Entities and features are problem specific
- We do not have a generic “pairing and features” PR
- You currently need to write your own

Feature examples

Features set	Description
tokens(6)	Surface string and POS for window of 6
type	Concatenated type of arguments
direction	Linear text order of arguments
distance	Sentence and paragraph boundaries
string	Surface string features of context
POS	POS features of context
intervening entities	Numbers and types of intervening entities
events	Intervening interventions & investigations

Performance by feature set

Feature set	P	R	F1
tokens(6) + type	33	22	26
+ direction	38	36	37
+ distance	50	70	58
+ string	63	74	68
+ POS	62	73	67
+ intervening entities	64	75	69
+ events	**65**	**75**	**69**
IAA			**47**
CIAA			**75**

End of Module 4