Machine Learning

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

Session Overview

- Introduction—what is machine learning & why do we want to do it?
  - Terminology
  - Development cycle
- Classification practical session
  - Training and application
  - Corpus QA classification metrics
  - Evaluating with cross-validation
- Chunking practical session
  - Training and application
  - Evaluating with Corpus QA and Annotation Diff
- Export and use of a corpus outside of GATE

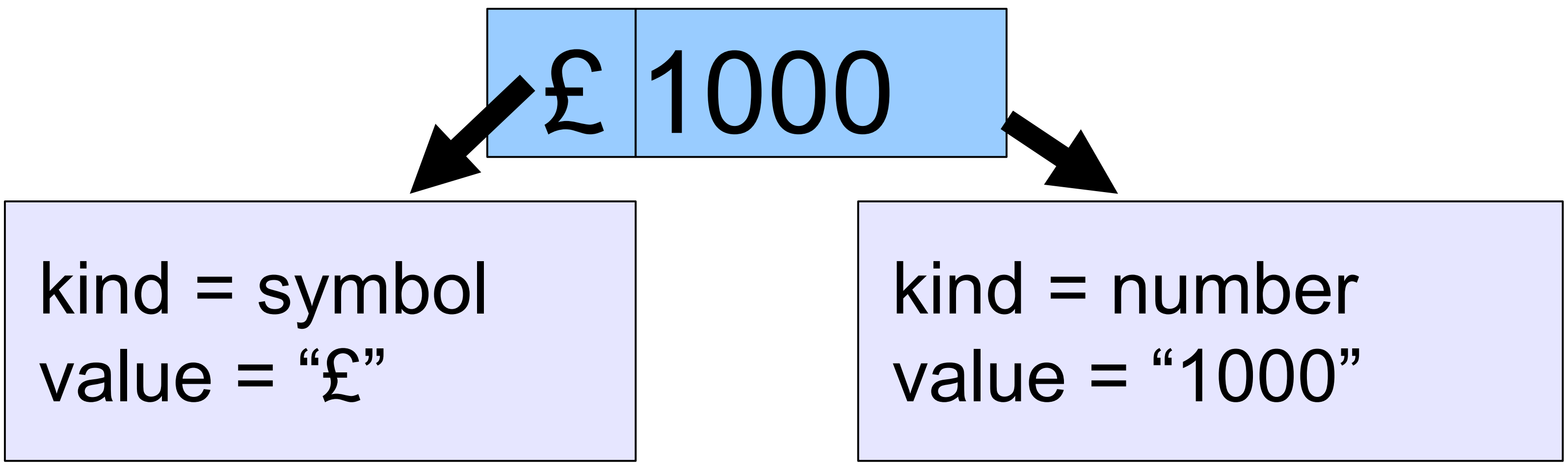
What is ML?

- Automating the process of inferring new data from existing data
- We will introduce ML by providing an overview of terminology only
- We cannot provide a tutorial on ML today due to limited time. However we'll introduce basic concepts in the context of NLP
- For a deeper insight, try:
  - Playing with Weka and reading the Weka book http://www.cs.waikato.ac.nz/ml/weka/index.html
  - Andrew Ng's course:

    https://www.coursera.org/course/ml

Learning a pattern

- In GATE, that means creating annotations by learning how they relate to other annotations
- 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?

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

Terminology

- Instances
- Attributes (or features)
- Classes

Instances

- Instances are cases that may be learned
- Every instance is a decision for the ML algorithm to make
- E.g. for each word in a sentence, what is it? Is it a location? Person? Neither? For each sentence instance, is it in French? English?

Attributes

- Attributes are pieces of information about instances
- They are sometimes called “features” in machine learning literature
- For example, the text string, its part of speech, the text of the preceding instance, all the words in the sentence ..
- When you want to identify e.g. persons in text, what information do you use?

Classes

- The class is what we want to learn
- Suppose we want to find persons' names: for every instance, the question is “is this a person name?” and the classes might be “yes” and “no”
  – (It's a touch more complicated—we'll get to that later)
- Sometimes there are many classes, for example we may want to learn entity types such as person, organization, location, date, ..
  – For every instance, the question is “which type from the list does this instance belong to?”

Terminology: Instances, attributes, classes

California Governor Arnold Schwarzenegger proposes deep cuts.

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

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

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

Classification tasks

- Opinion mining
  - Example: the documents contain spans of text (such as individual sentences or longer consumer reviews) which you want to classify as positive, neutral, or negative
- Genre detection: classify each document or section as a type of news
- Author identification
- Classifying sentences according to language

Classification tasks

- Imagine you are classifying sentences according to whether they express a positive or a negative opinion
- What would the instance be?
- What might be useful attributes?
- What would the classes be?

Example: text classification

The	new	Acme	Model	33	stinks	!
Token	Token	Token	Token	Token	Token	Token
		Product				
Sentence (polarity=negative)						

- instance: Sentence annotation
- attributes: Token and Product annotations and their features (suppose that the Product annotations have been created earlier with gazetteers and rules)
- class: polarity= "negative"

- ML could learn that a Product close to the Token "stinks" expresses a negative sentiment, then add a polarity="negative" feature to the Sentence.

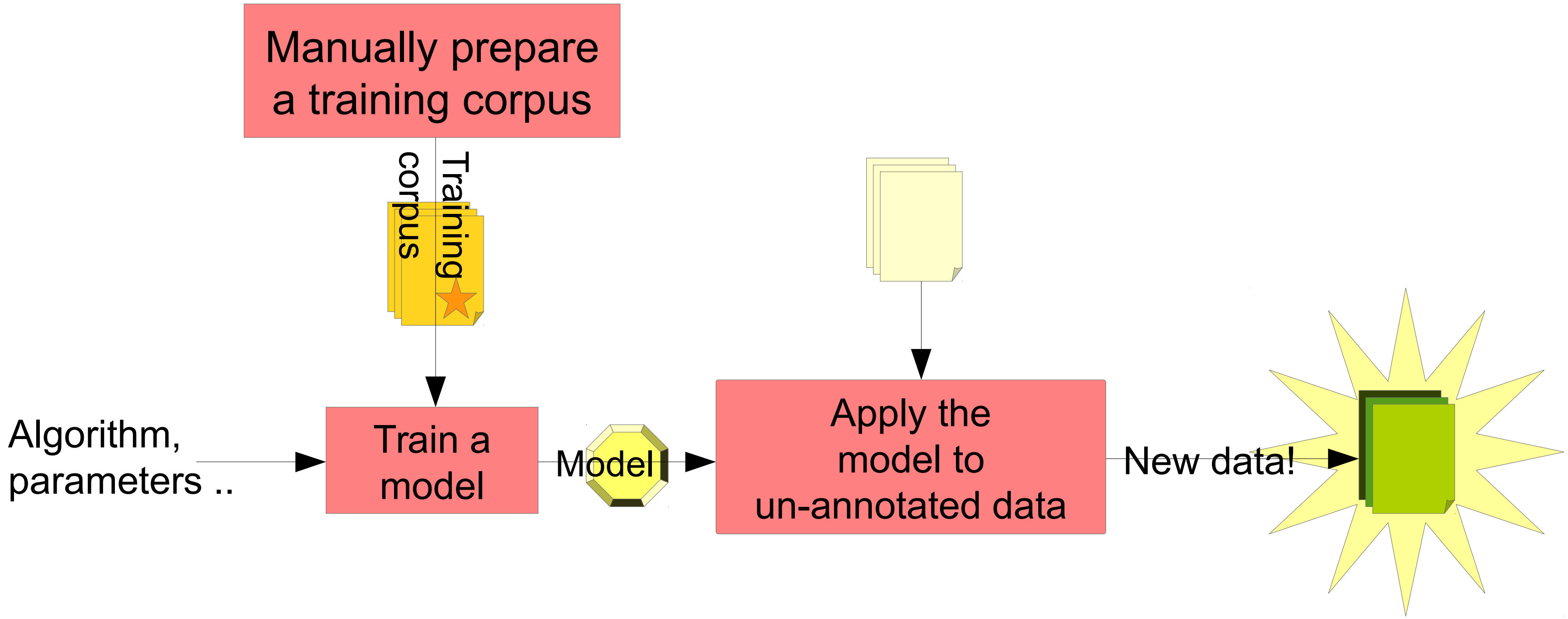
Training

- 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

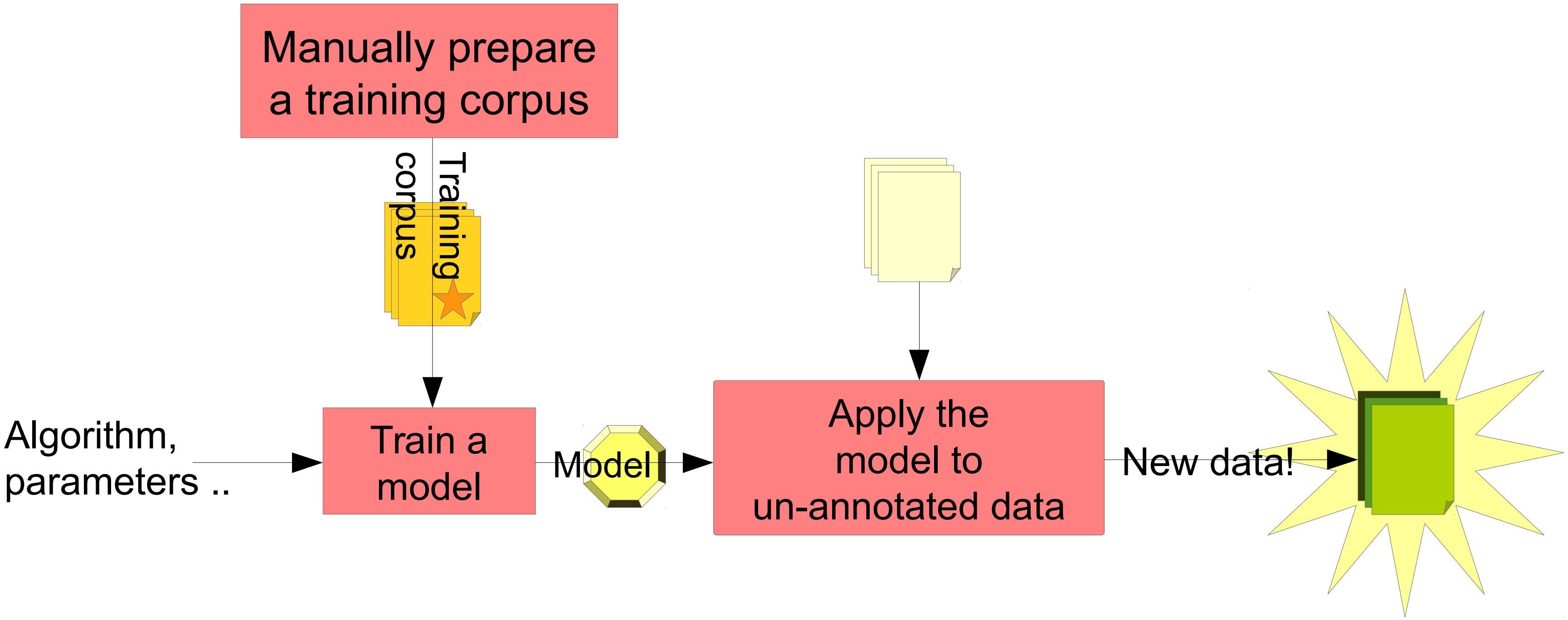
Application

- 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

Development Cycle



Development Cycle

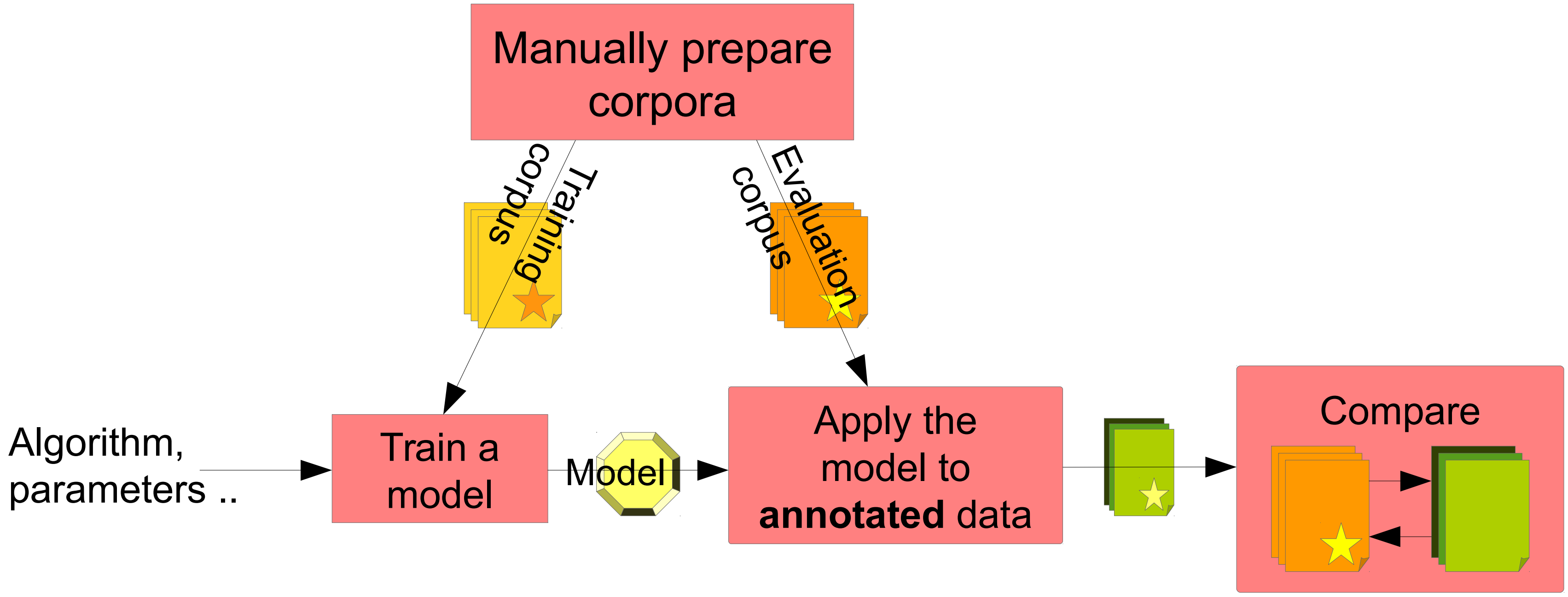


But how do we know how good it is?

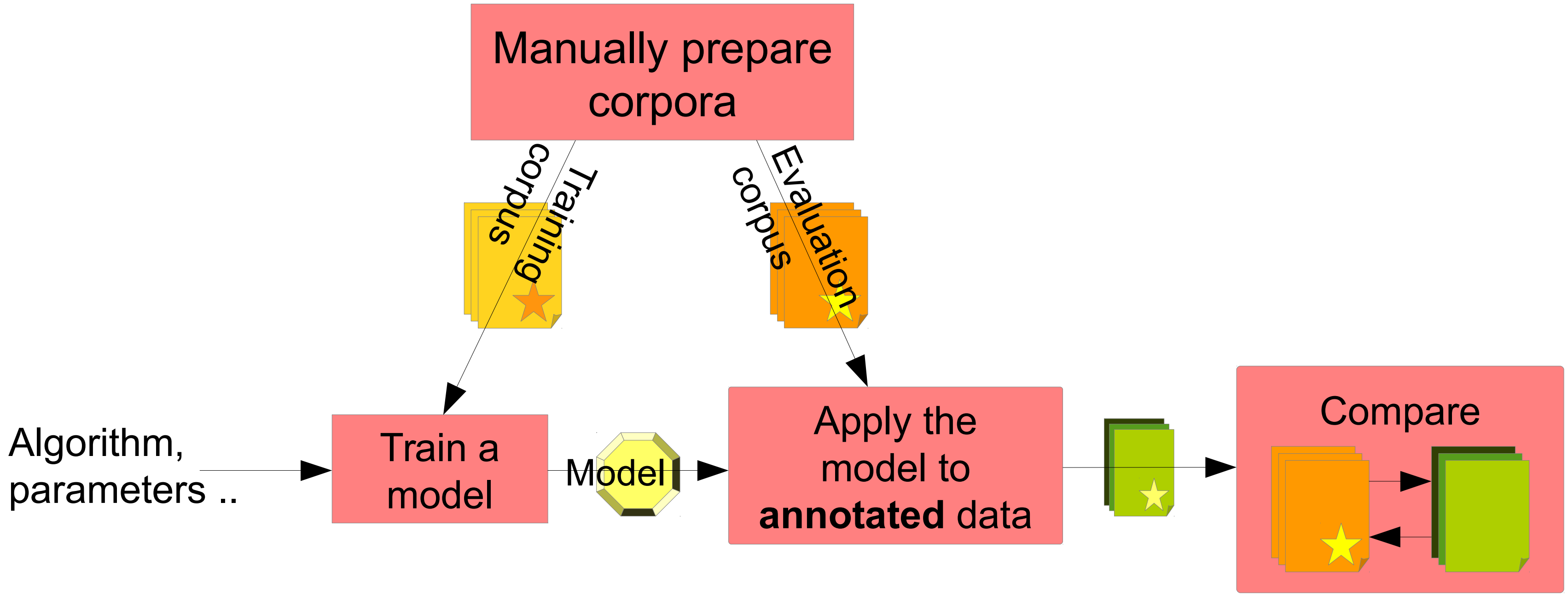
Evaluation

- 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

Development Cycle



Development Cycle

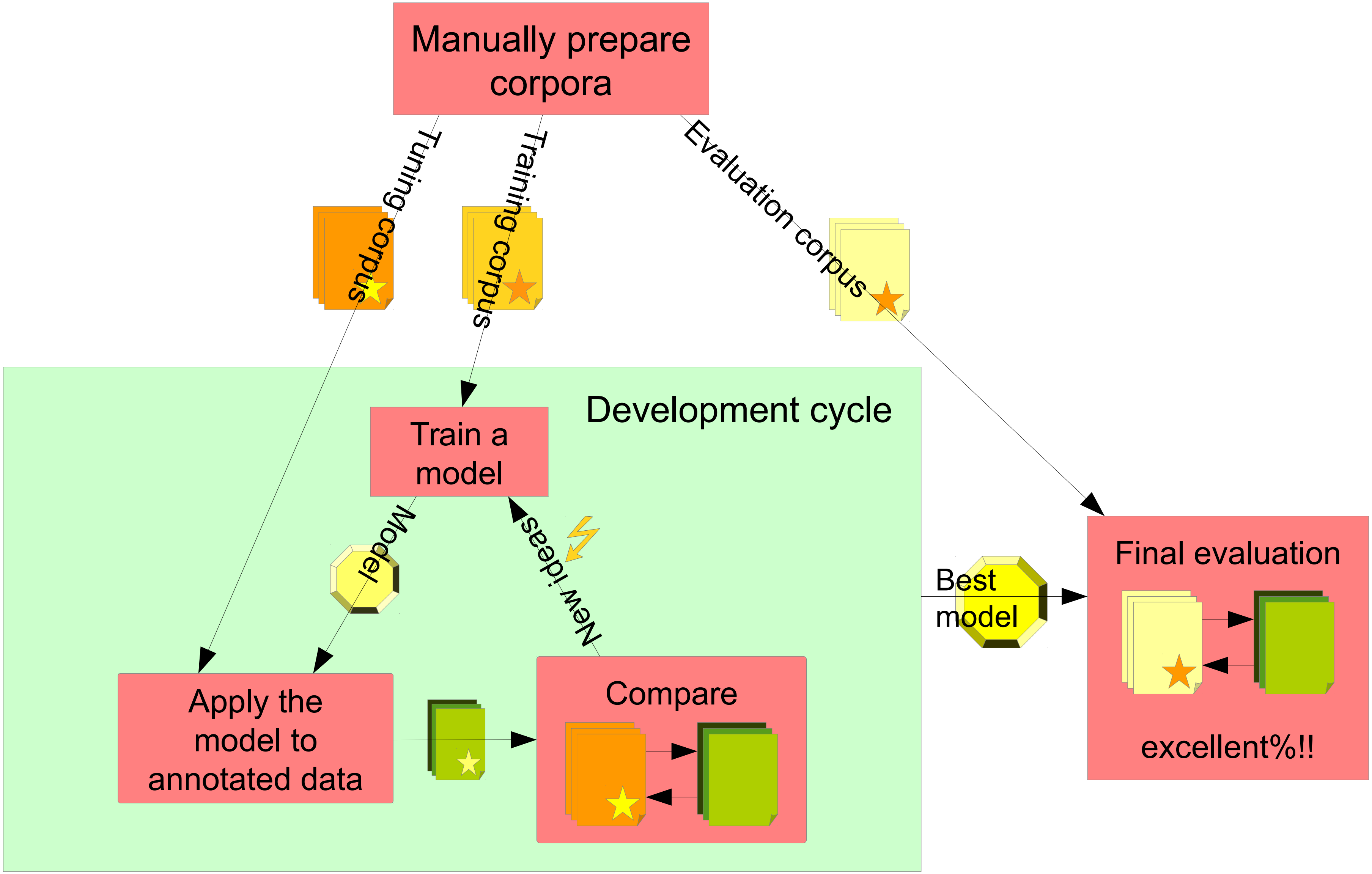


But I don't like that result! I want to make it better!

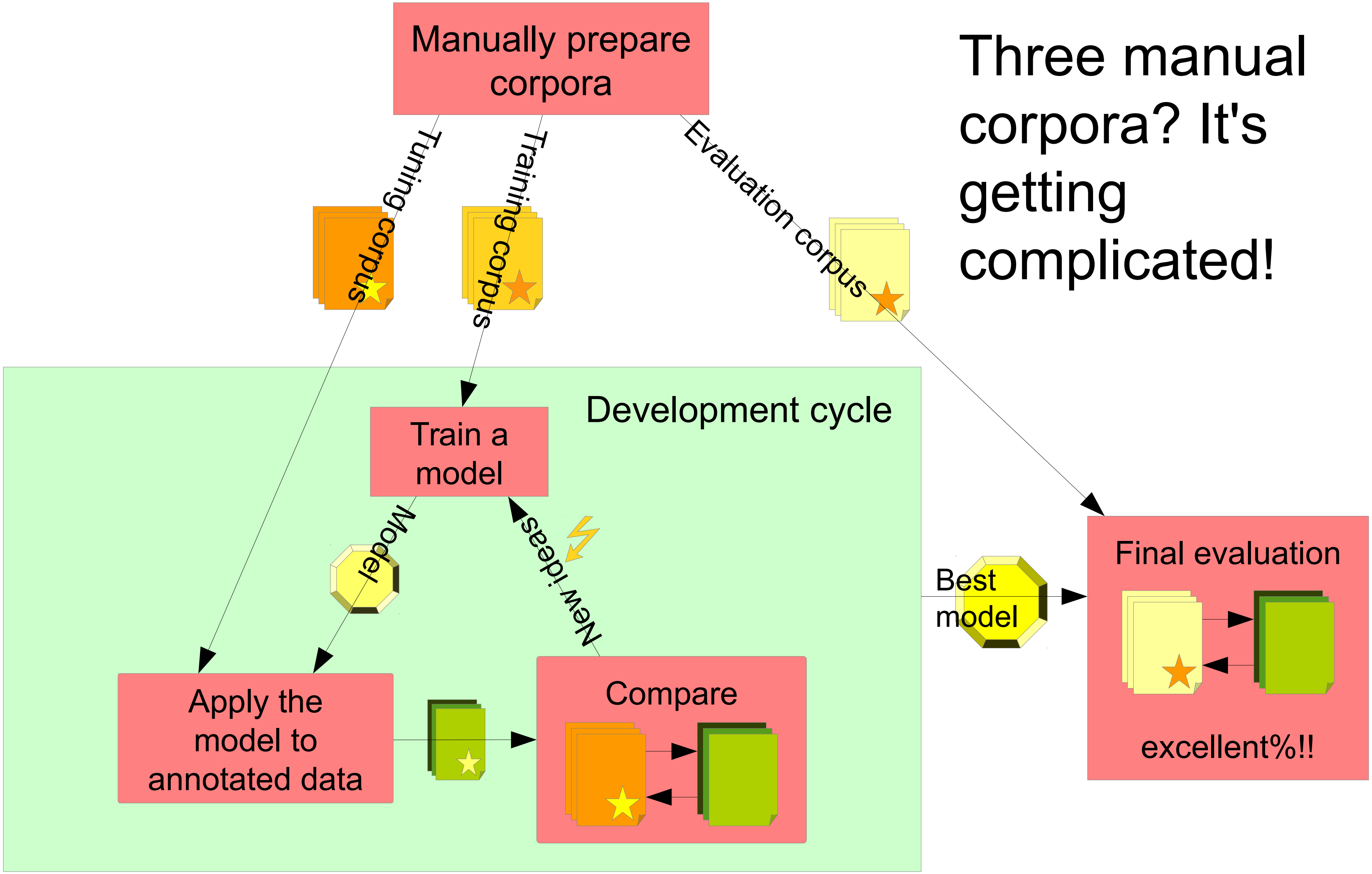
Tuning

- An important part of machine learning work is trying different things to get a good result
- However, be aware that if you tune to get a good result on a corpus, it will be artificially good!
- Some of what you learned transfers to new data, but some of what you learned may be specific to this corpus
- So you need a fresh corpus to get a final result

Development Cycle

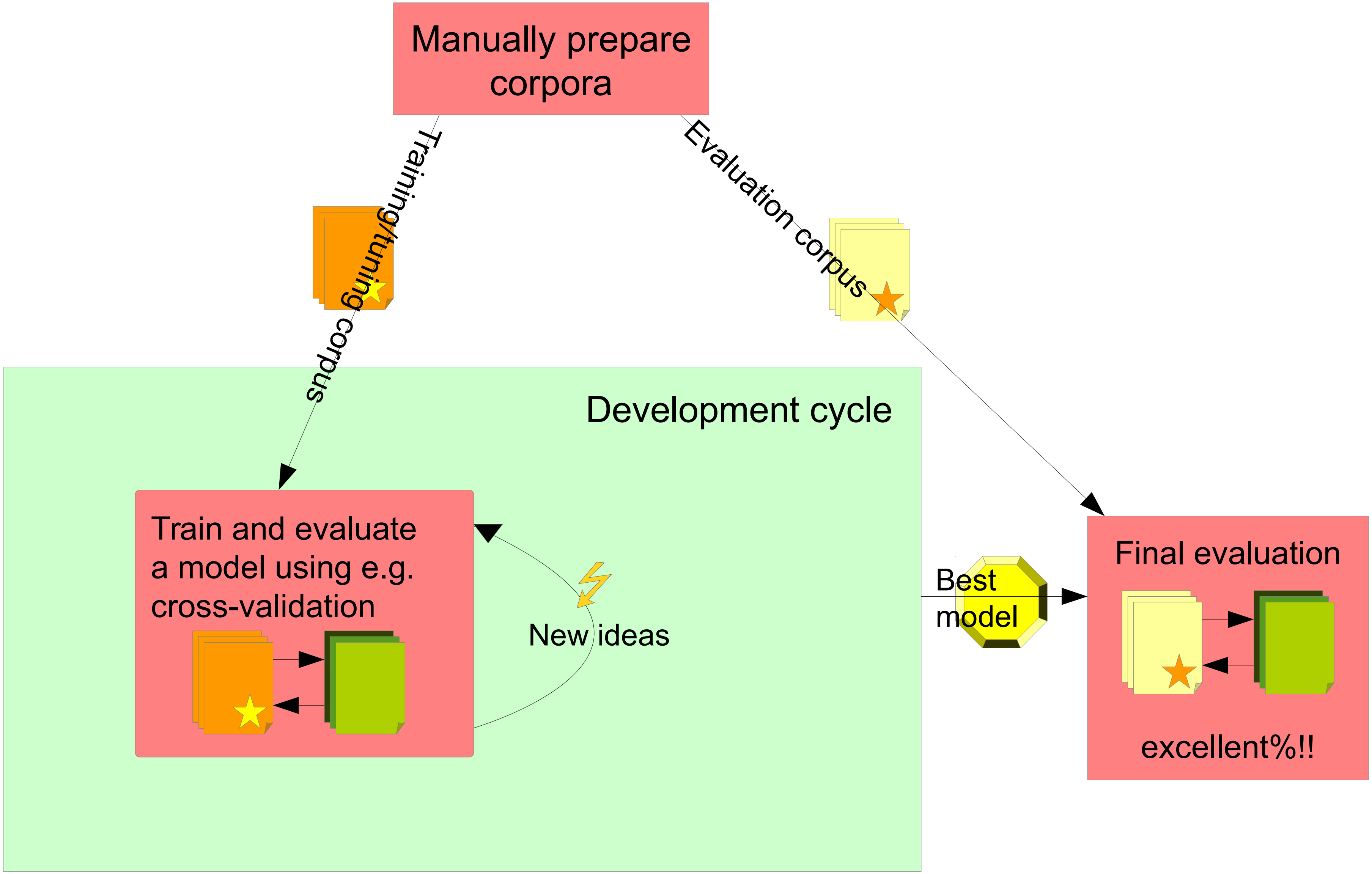


Development Cycle



Three manual corpora? It's getting complicated!

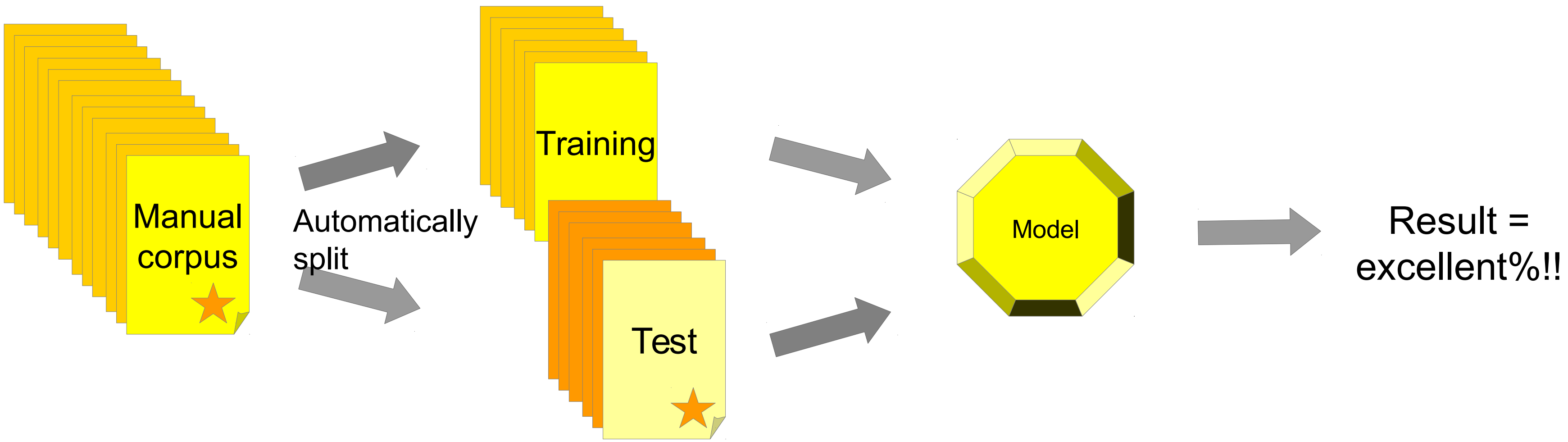
Development Cycle



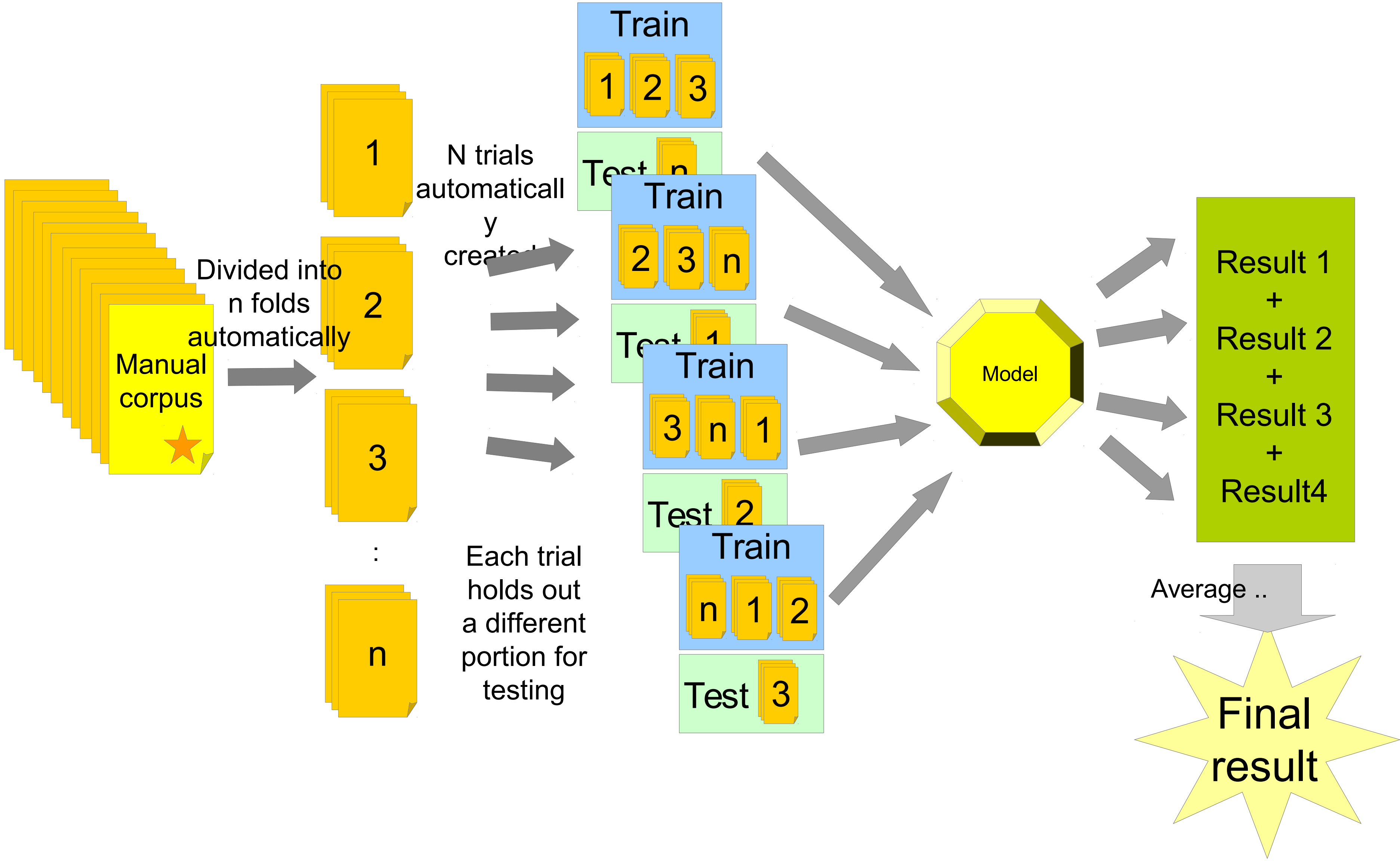
Cross-validation and hold-out evaluation

- The process of splitting a corpus for training and application can be facilitated, so you don't have to split the corpus and run separate training and application steps yourself
- Hold-out evaluation holds back a portion of the corpus for testing
- You can automatically do this a number of times and take an average
- Cross-validation splits the corpus into n portions ("n-fold cross-validation) and in turn, holds each out for testing, then averages all the results
- You could hold out just a single instance each time, maximizing your training portion! The more folds, the longer it takes though
- All you have to do is select which you want, and everything is done automatically

Hold-out evaluation



N-fold cross-validation



Machine Learning in GATE

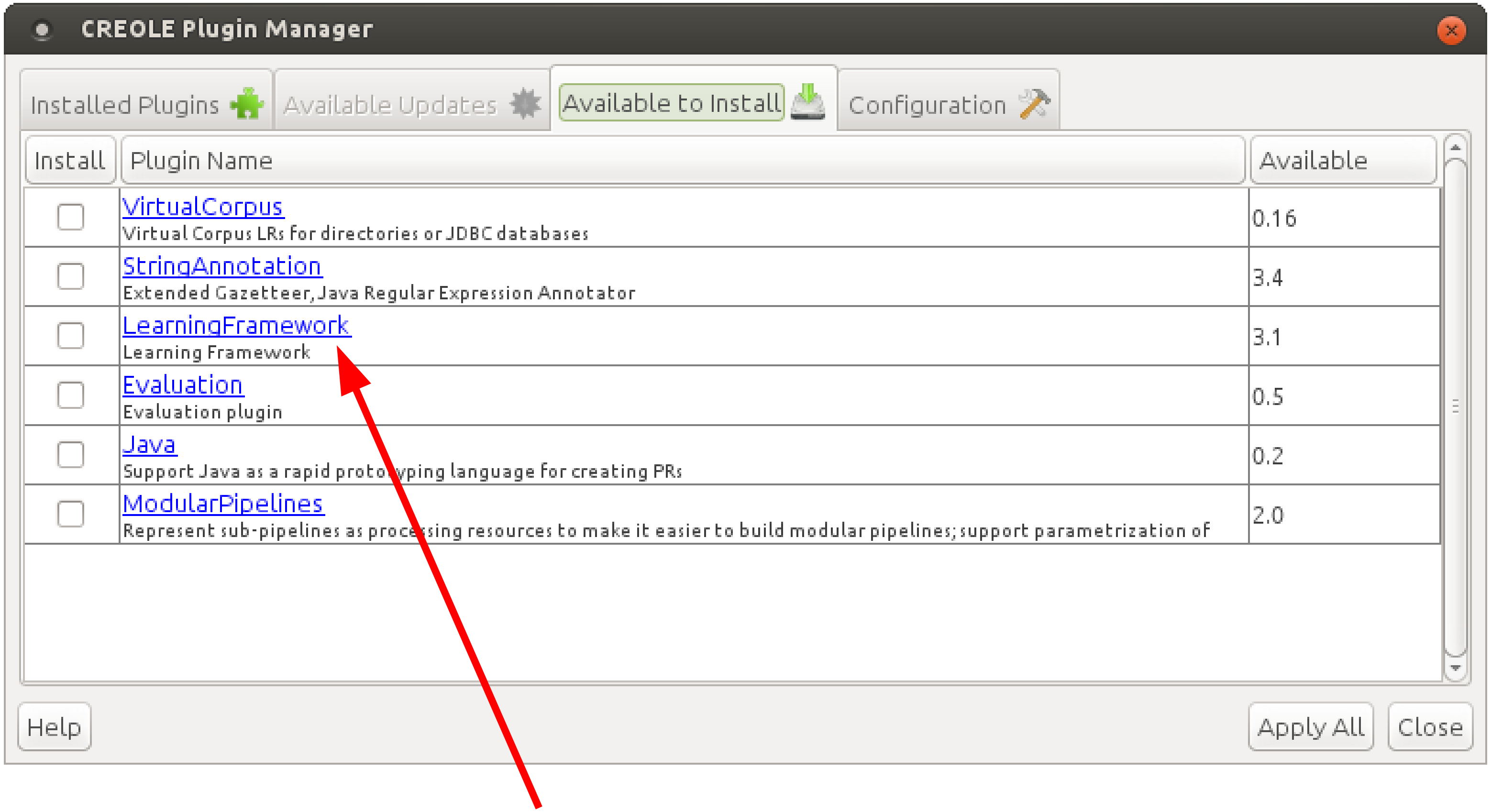
- GATE supports machine learning in several ways
- Some of the **standard PRs** are ML-based e.g. Stanford parser
- **Third-party NLP components**
  - e.g. the OpenNLP PR can be used with any models, trained externally to GATE
- **Dom's Python PR** makes it easy to hook up to any Python process
  - https://github.com/GateNLP/gateplugin-python
- **Batch Learning PR** and **Machine Learning PR**: Old and older(!) GATE ML PRs. Batch Learning PR has been the main GATE ML offering for many years, but isn't going to be supported any more. Machine Learning PR is was our first ML integration.
- **Learning Framework**
  - Integrates more libraries, including Mallet's CRF
  - Export to ARFF and compatible algorithm availability allows feature selection and parameter tuning in Weka
  - Relatively easy for a Java programmer to extend with any further algorithms they need (and send us a pull request!)

Getting the Learning Framework Plugin

- You need to make a directory to store your plugins in, and indicate this in the configuration tab of the plugin manager
- Then select "plugins from the GATE team" in the same tab
- The plugin will then be available to install in the "available to install" tab
- Alternatively you could use Git to clone it from here into your user plugins directory:

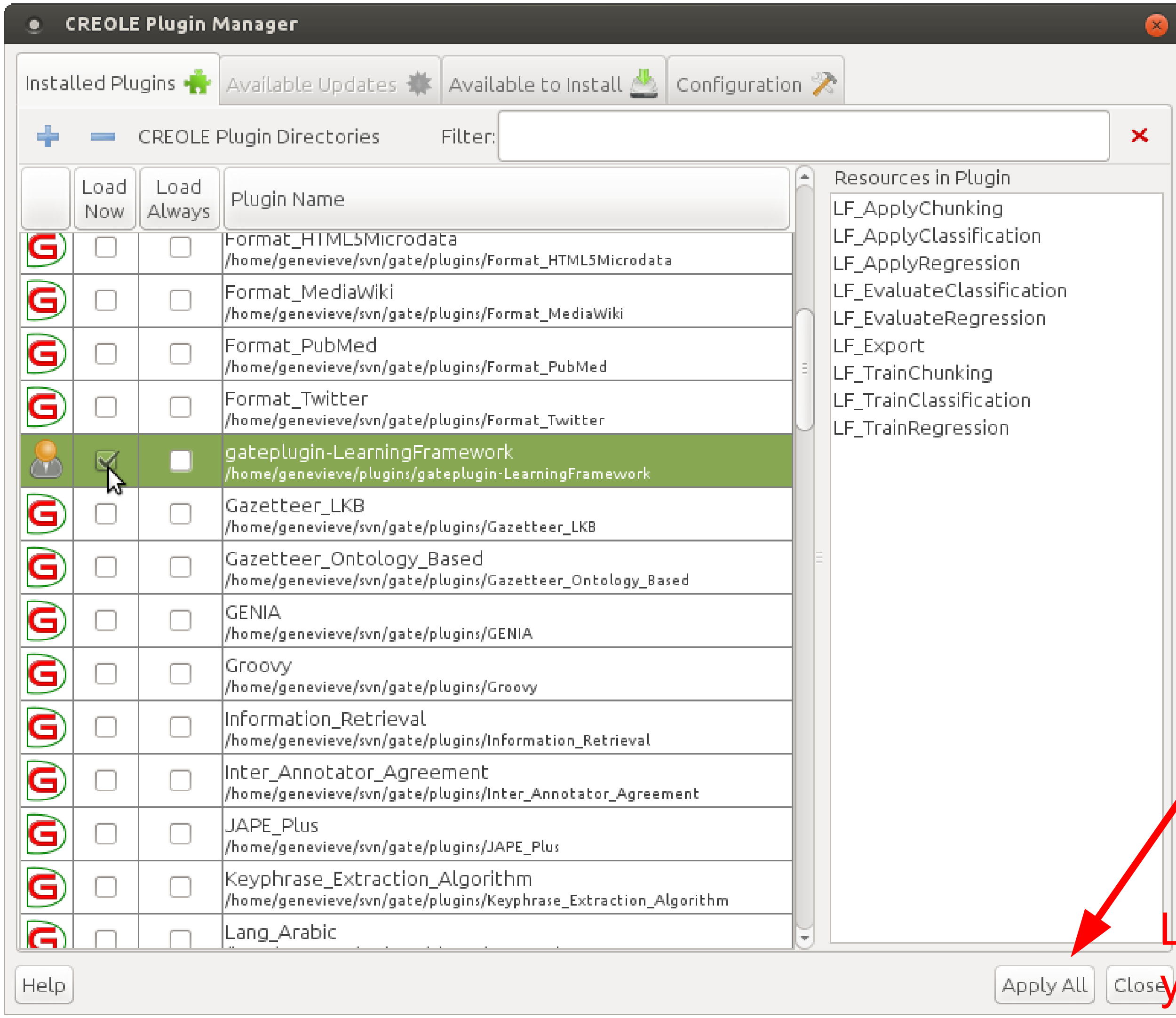
  https://github.com/GateNLP/gateplugin-LearningFramework
- Then you would need to build it using Ant

Getting the Learning Framework Plugin



On the “Available to Install” tab, select Learning Framework version 3.1

Getting the Learning Framework Plugin

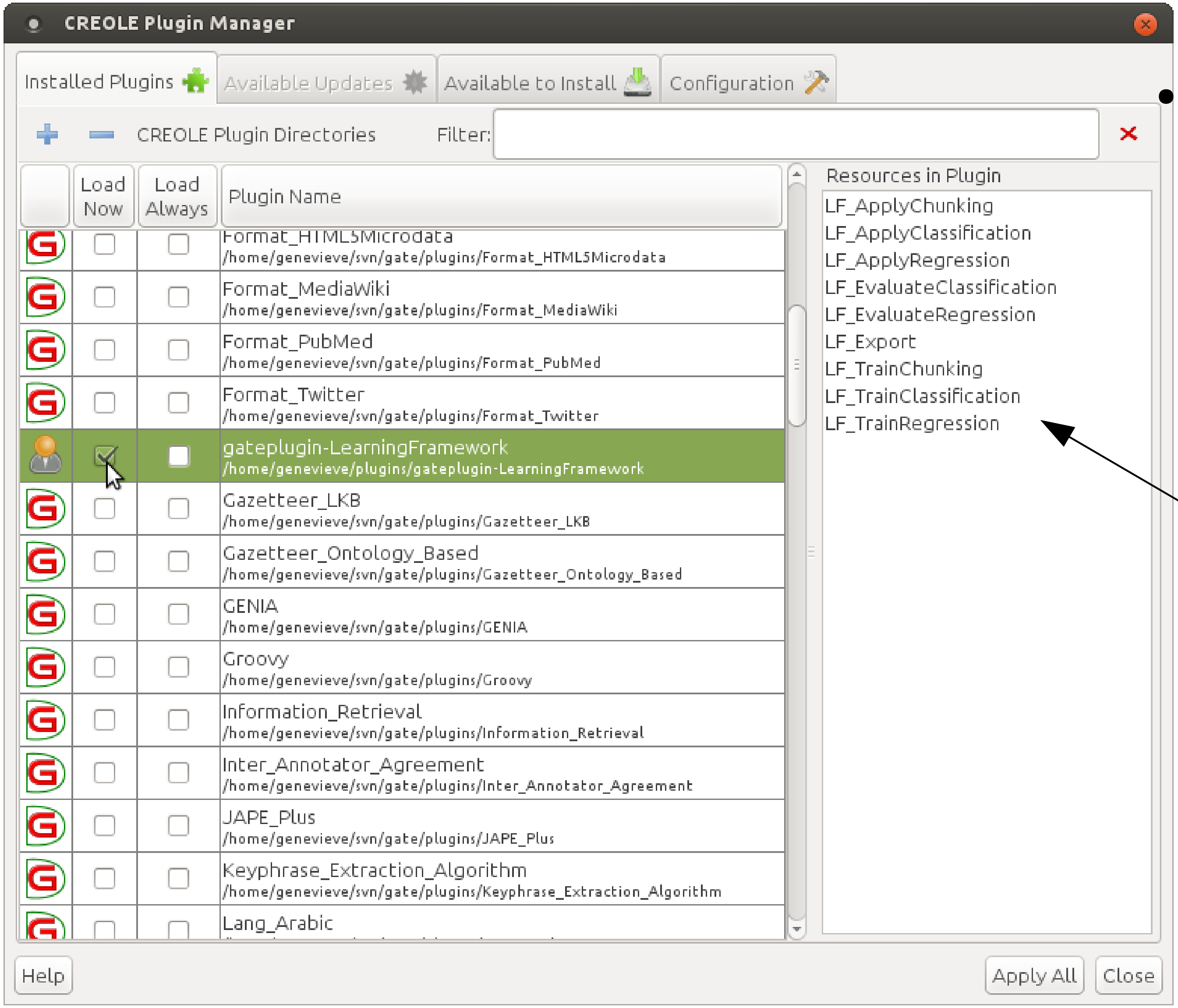


Load the Learning Framework plugin. It starts with a “g”!!!

Don't forget to apply!

Load the Tools PR, while you're there, if you haven't already!

PRs in the plugin



- In the plugin manager you might have noticed that the Learning Framework plugin contains nine PRs

ML Tasks in the Learning Framework

- The Learning Framework supports 3 types of ML tasks:
- Chunking (named entity recognition, finding NPs)
- Classification (sentiment classification, POS tagging)
- Regression (assigning a number rather than a type, for example ranking candidates for named entity linking)
- Separate PRs for training and application facilitate each of these tasks

PRs in the Plugin

- Evaluate Classification PR provides an accuracy figure for classification evaluation (cross-validation and hold-out)
  - Can be used to evaluate the classification aspect of chunking—more on this later
  - Evaluate Chunking PR is forthcoming .. But in the mean time you can use the normal GATE evaluation tools
- Export—data are exported for use in ML tools outside of GATE

Documentation

- The documentation for the plugin is available here:

  https://github.com/GateNLP/gateplugin-LearningFramework/wiki
- You can find advice about algorithm parameters, feature specification and so on
- In today's course you will be asked to use your initiative at times, and may find it useful to consult this wiki!

Classification—Practical Exercise

Classification—Practical Exercise

- Materials for this exercise are in the folder called "classification-hands-on"

Classification using Training and Application PRs

Load the corpus

- Create a corpus for testing and one for training (make sure you name them so you can tell which is which!)
- Populate them from classification-hands-on/test-corpus and classification-hands-on/training-corpus
- Open up one of the documents and examine it

Examining the corpus

- The corpus contains an annotation set called “Key”, which has been manually prepared
- Within this annotation set are sentences with a “lang” feature indicating the language of the sentence

What are we going to do with this corpus?

- We are going to train a machine learner to annotate sentences with their language
- We'll start with separate training and application steps
- Later we can try some of the evaluation techniques we talked about earlier

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 parts of ANNIE over the corpus, then we can use:
  - Sentence annotations for instances
  - Token features for attributes

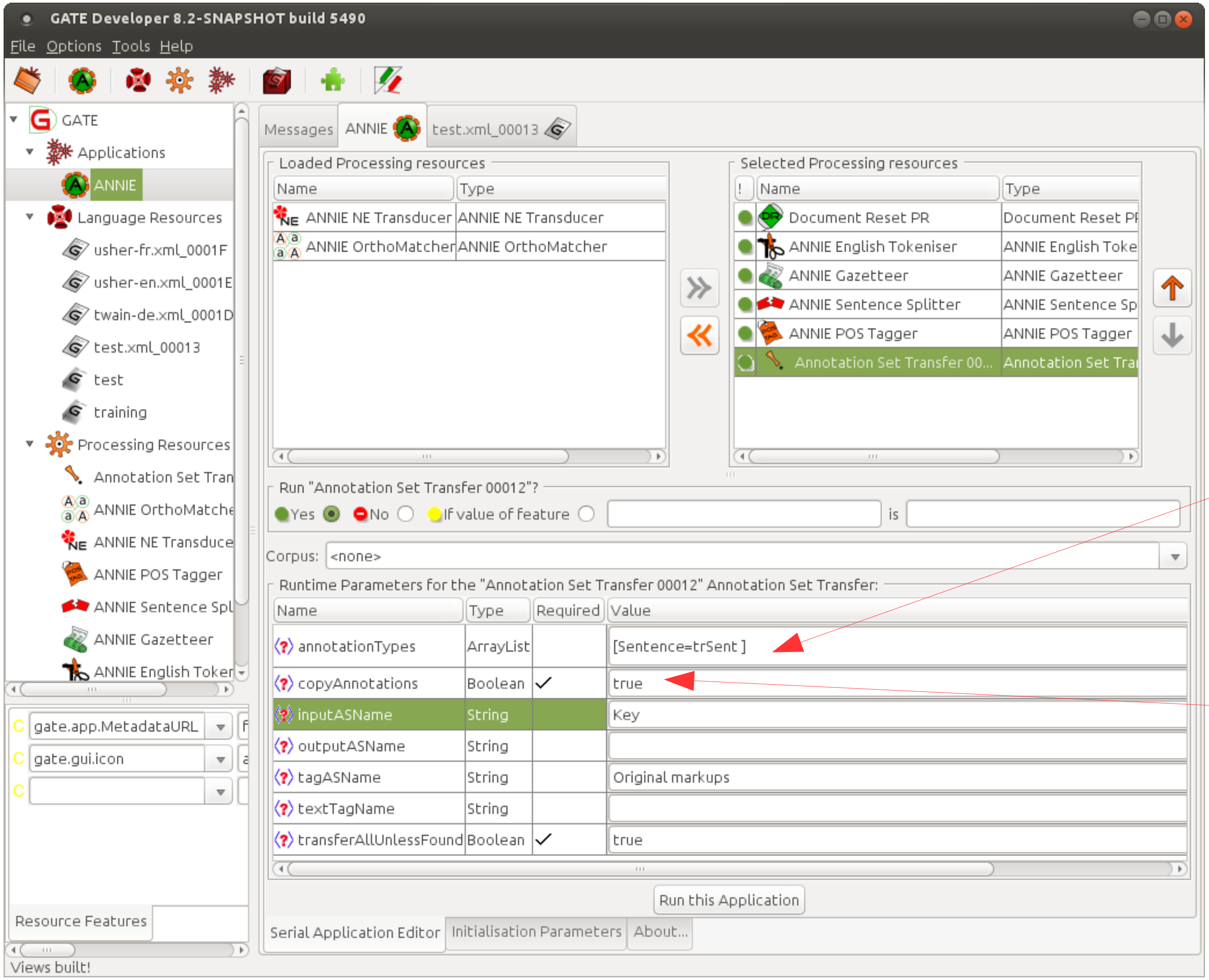
Making the Application

- Load ANNIE with defaults
- We only want tokens and some basic features so remove the last two PRs from the pipeline
  - ANNIE NE Transducer
  - ANNE Orthomatcher
- Check that the document reset PR's setsToKeep parameter includes “Key”!

Annotation Set Transfer

- The Learning Framework expects all class and feature annotations to be in the same set
- ANNIE puts annotations in the default set
- So we need to copy the sentences from Key into the default set
    - (We could have ANNIE output to “Key” but it would be a lot more hassle, and “Key” should be reserved for manual annotations really)
- We can use the Annotation Set Transfer PR to do this
- However, ANNIE also makes sentence annotations! To avoid confusion, we'll call these gold standard sentences something different

Annotation Set Transfer



- Create an Annotation Set Transfer PR (if you can't find it, perhaps you forgot to load the Tools plugin)
- Add it to your application
- "=" notation in the copyAnnotations parameter allows us to rename the annotation type
- Be sure to "copyAnnotations"!!!!

Training PR

- Make a PR for classification training and add it to the application at the end
- Make one for application too—we'll come to that later. Don't add it yet though

Training PR—Parameters

- algorithmParameters—parameters influencing the algorithm, documented either in the library's own documentation or LF documentation on GitHub
- dataDirectory—where to put the model (it will be saved as a Java object on disk). It should be a directory that already exists.
- featureSpecURL—The xml file containing the description of what attributes to use
- inputASName—Input annotation set containing attributes/class
- instanceType—annotation type to use as instance

Training PR—Parameters

- scaleFeatures—use a feature scaling method for preparation? Some algorithms prefer features of similar magnitude (advanced)
- sequenceSpan—for sequence classifiers only. We'll look at this in the context of chunking
- targetFeature—which feature on the instance annotation indicates the class
- trainingAlgorithm—which algorithm to use

Feature Specification

```
<ML-CONFIG>

<NGRAM>
<NUMBER>1</NUMBER>
<TYPE>Token</TYPE>
<FEATURE>string</FEATURE>
</NGRAM>

</ML-CONFIG>
```

- This file is in your hands-on materials
- Feature specification indicates which attributes we are going to use
- This one just uses the strings of the tokens
- What else might be useful for identifying the language a sentence is written in?

Feature Scaling

- Feature scaling is an advanced feature that we won't make use of today
- However it can be essential to getting a good result!
- Behind the scenes, all features are converted into numbers, for example one for the presence of a word or zero for its absence
- Other features might be the length of a word, which might range from one to twenty or more, or a frequency figure that might be a very large number
- Many algorithms work better if features have the same approximate magnitude
- Therefore after features have been gathered from the corpus, it can make sense to scale them

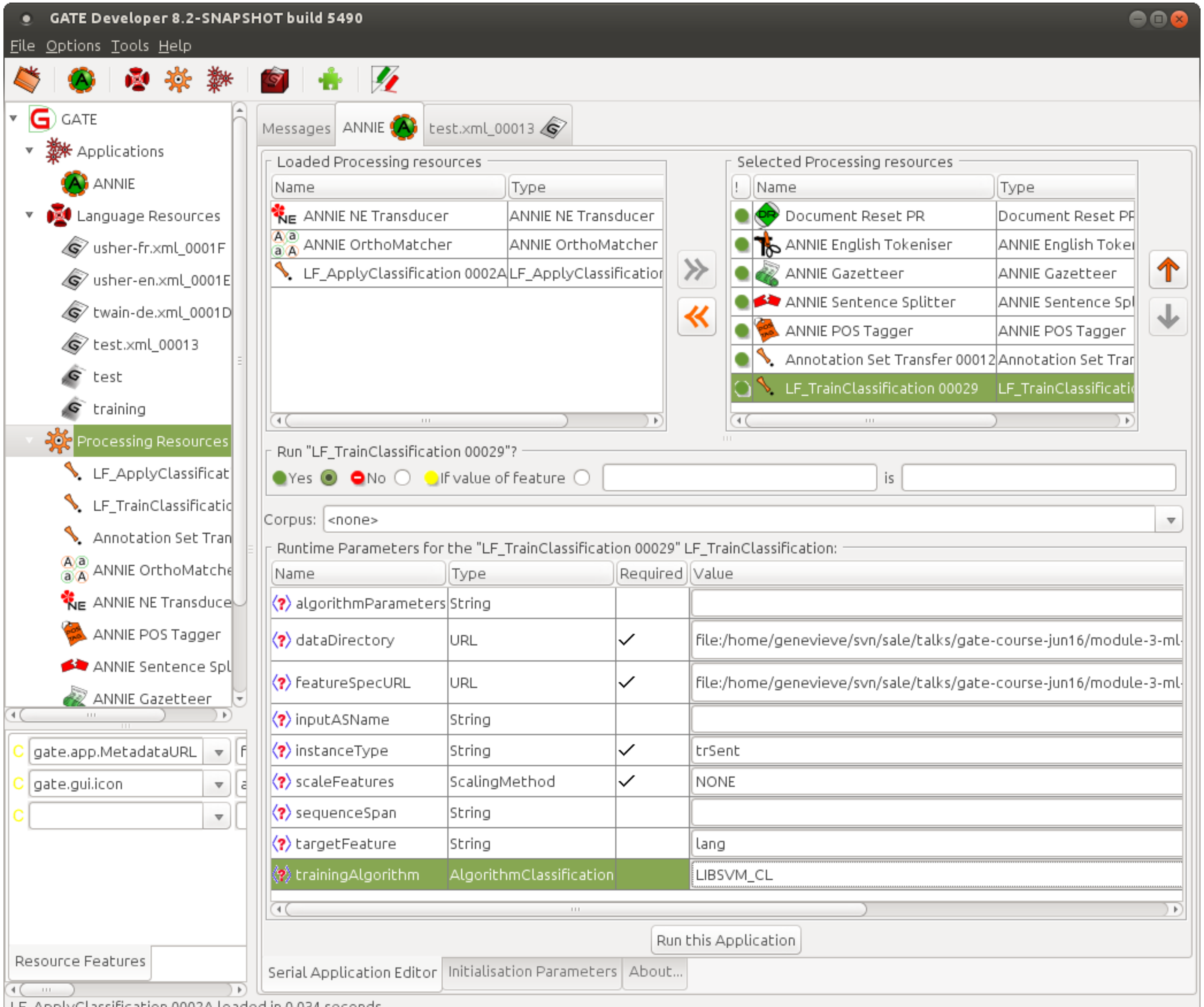
Algorithms

- Three libraries are integrated/available; Mallet and Weka, each providing many algorithms, and LibSVM (support vector machine)
- Weka requires a separate download
- Names begin with the library they are from
- After that, “CL” indicates that it's a classification algorithm and “SEQ” indicates a sequence learner
- Where to start?
  - SVM is good but you must tune it properly
  - Decision trees can be interesting to read
  - (Weka wrapper—Random Forest is good)
  - CRF is good for chunking
  - Try a few and see for yourself!

Set parameters for training

- Be sure to set the dataDirectory to a place you can store your trained model; perhaps the hands-on folder for this classification exercise?
  - Unlike the evaluation PR, training creates a persistent model on disk that you can reuse later
  - The application PR will use the model it finds there
- You need to set the targetFeature to “lang” (why?)
- For algorithm, let's try LibSVM
- Set the feature spec URL to point to the feature XML file “classification-features.xml” in your hands on materials
- instanceType should be whatever you created with your AST

Training Classification



- Be sure to choose the right corpus for training
- Go ahead and train your model!

Training a model

- Switch to the messages pane so you can see the output
- Did it look like it worked? Can you find where it tells you what classes you have and how many features? Does it look right to you?

Classification Application

- Move the training PR out of the application, and put the application one in instead
- You can also take out the Annotation Set Transfer
  - We don't need the right answers at application time!
  - They can stay where they are, in Key, and we'll use them to compare with our new ML annotations later

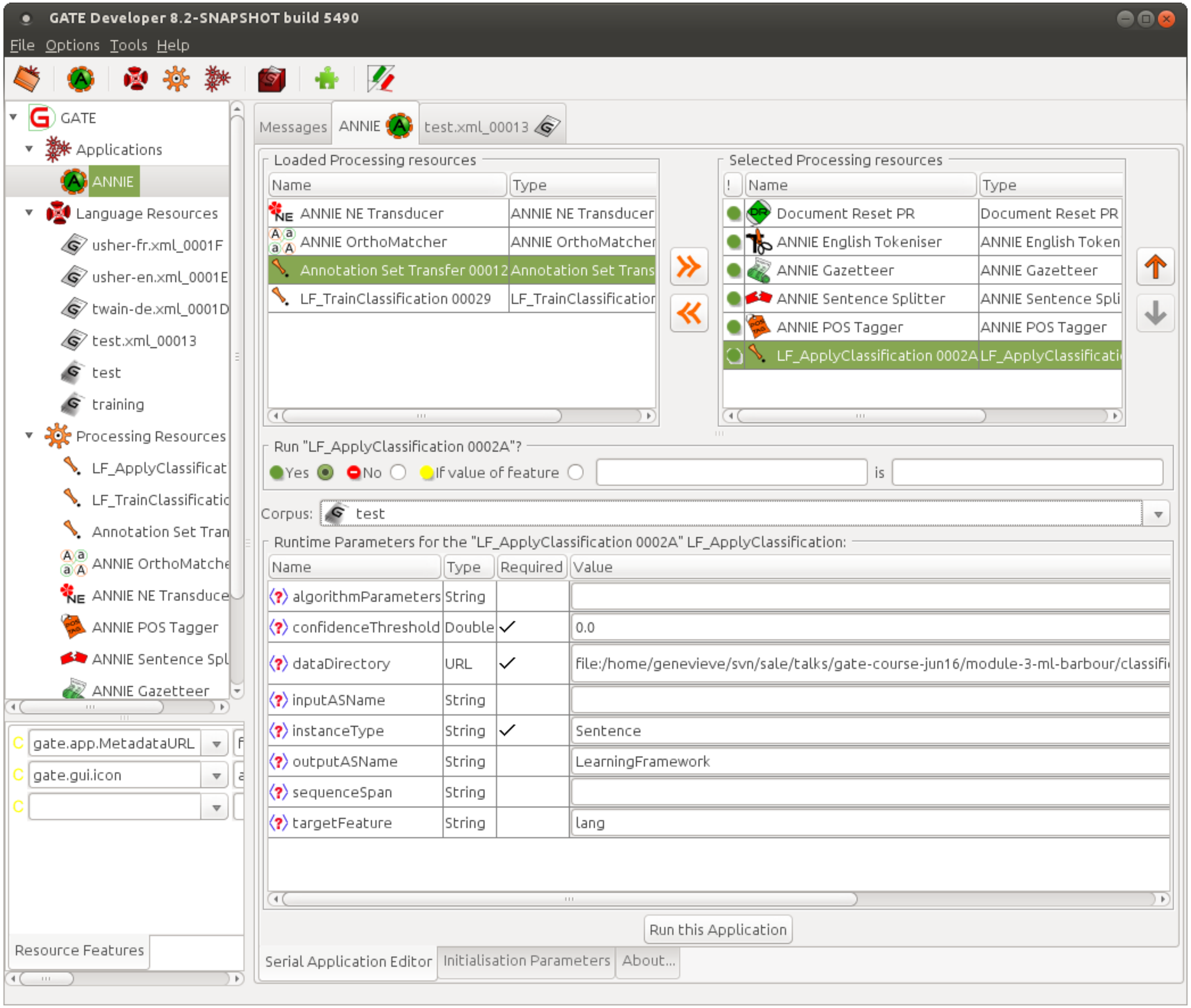
Classification Application

- Many of the parameters are the same as for the training PR
- outputASName indicates where the final answers will go
  - If you set it blank, the classes will go back onto the instances
  - If you're applying to a test set, this may overwrite your class feature! So be careful! Though in our case, the class is in Key
  - The default of “LearningFramework” is fine
- Set instanceType
  - At training time, we learned from the Key annotations
  - At application time, we can just classify the sentences that ANNIE found for us
  - So what do you think instanceType should be?

Classification Application

- You can set dataDirectory as previously, so it can find the model you just trained
- targetFeature needs to be the same as the one in the Key set, so that when we evaluate it matches
- confidenceThreshold allows you to set a threshold for how certain the model needs to be to assign a class. For a well tuned model it shouldn't be necessary. It's more relevant for problems such as finding named entities (more on that later). So we'll leave it blank

Applying a model



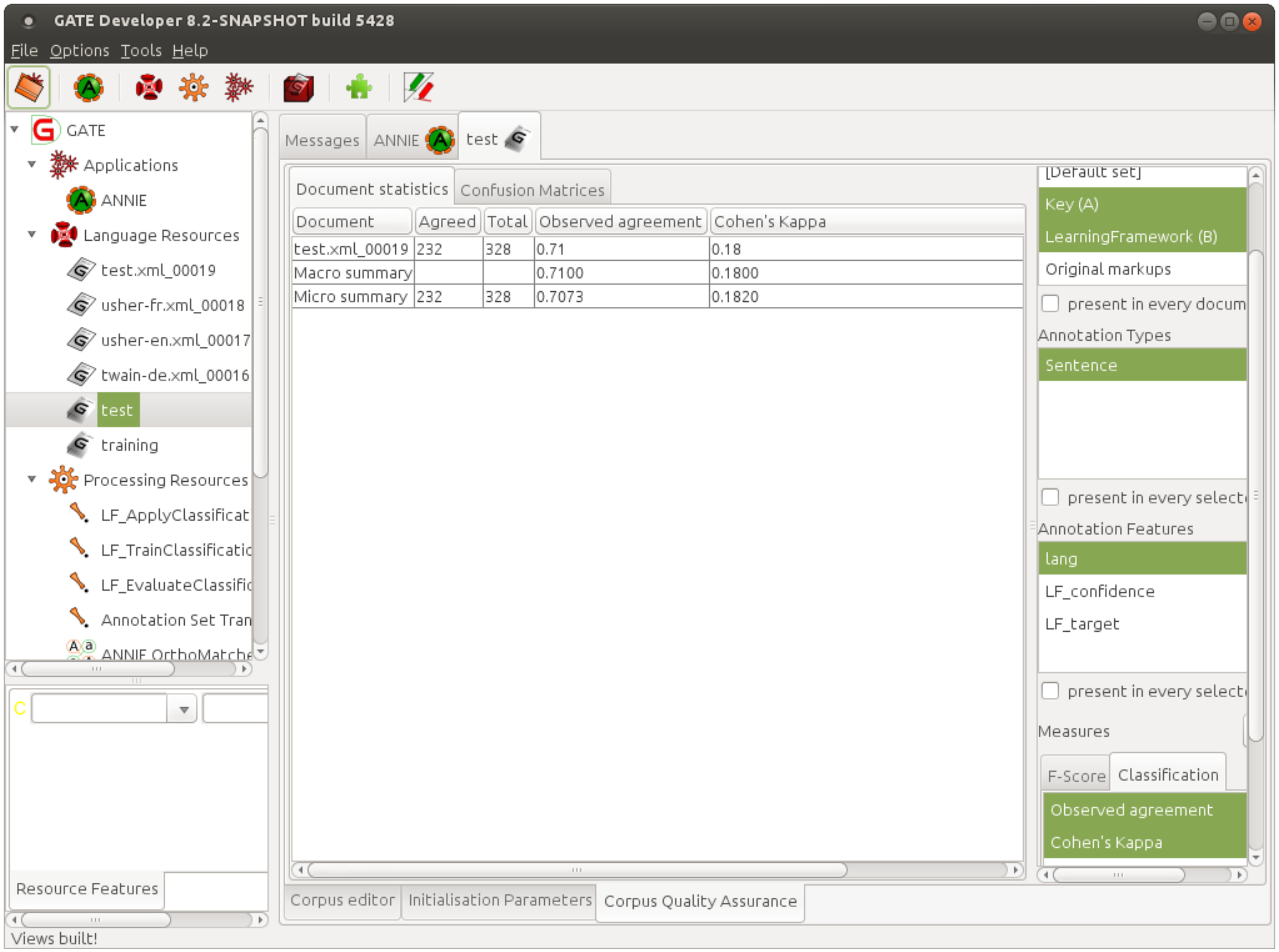
- Make sure you have selected the test corpus
- Go ahead and run the application!

Examining classification results using Corpus QA

Evaluating Classification

- Accuracy is a simple statistic that describes how many of the instances were correctly classified
- But what constitutes a good figure? 95%
- What if 99% of your instances are the majority class? You could get an accuracy of 99% whilst completely failing to separate the classes and identify any of the minority class instances at all!
- Kappa metrics provide a measure of the statistical independence of your result from the actual right answers
- Accuracy is a useful metric for parameter tuning but tells you little about how well your system is performing at its task

Corpus QA for classification



- In the Corpus QA tab, select annotation sets to compare, instance type and class feature and choose both agreement and a kappa statistic
- Click on “Compare”

Classification metrics

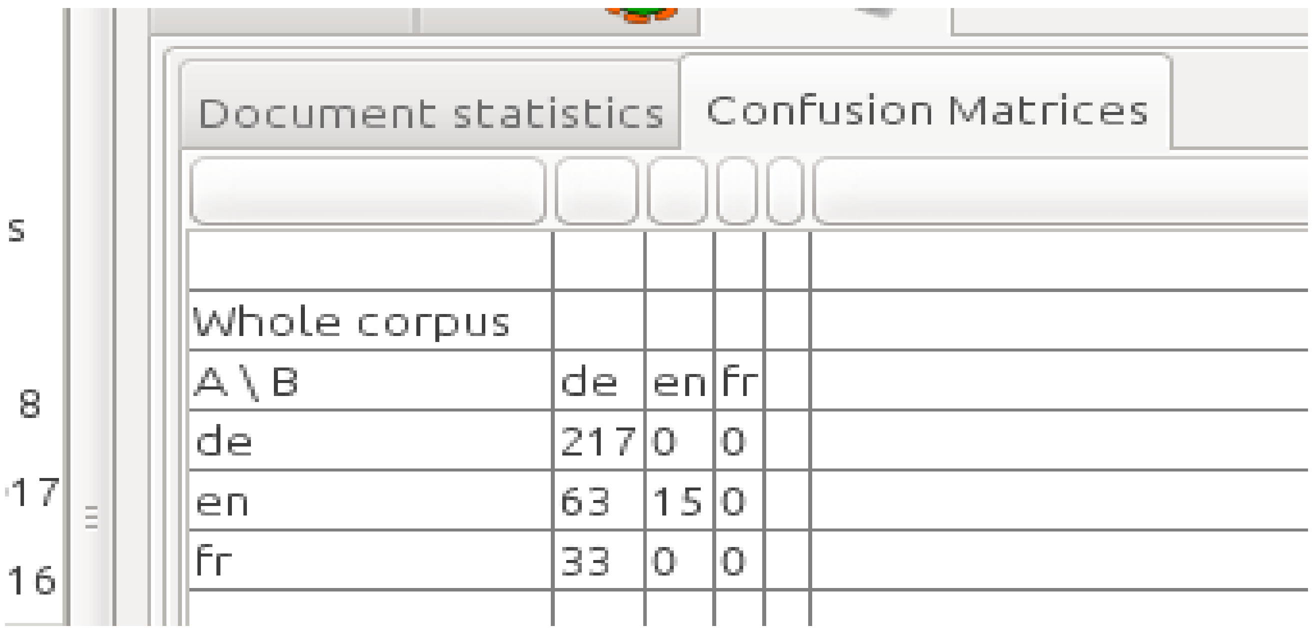
- What do you think about this result? Not bad?
- What do you think of this kappa statistic? (A kappa of over 0.5 is considered good, and over 0.8 excellent.)

Confusion matrices

- Often you can learn a lot about what might be improved by looking at the kind of mistakes your classifier is making
- A confusion matrix shows you which types tend to get confused with which other types

Confusion Matrices

- Confusion matrices are available on the next tab (at the top of the screen)
- What do you think about the misclassifications?



Classification Evaluation

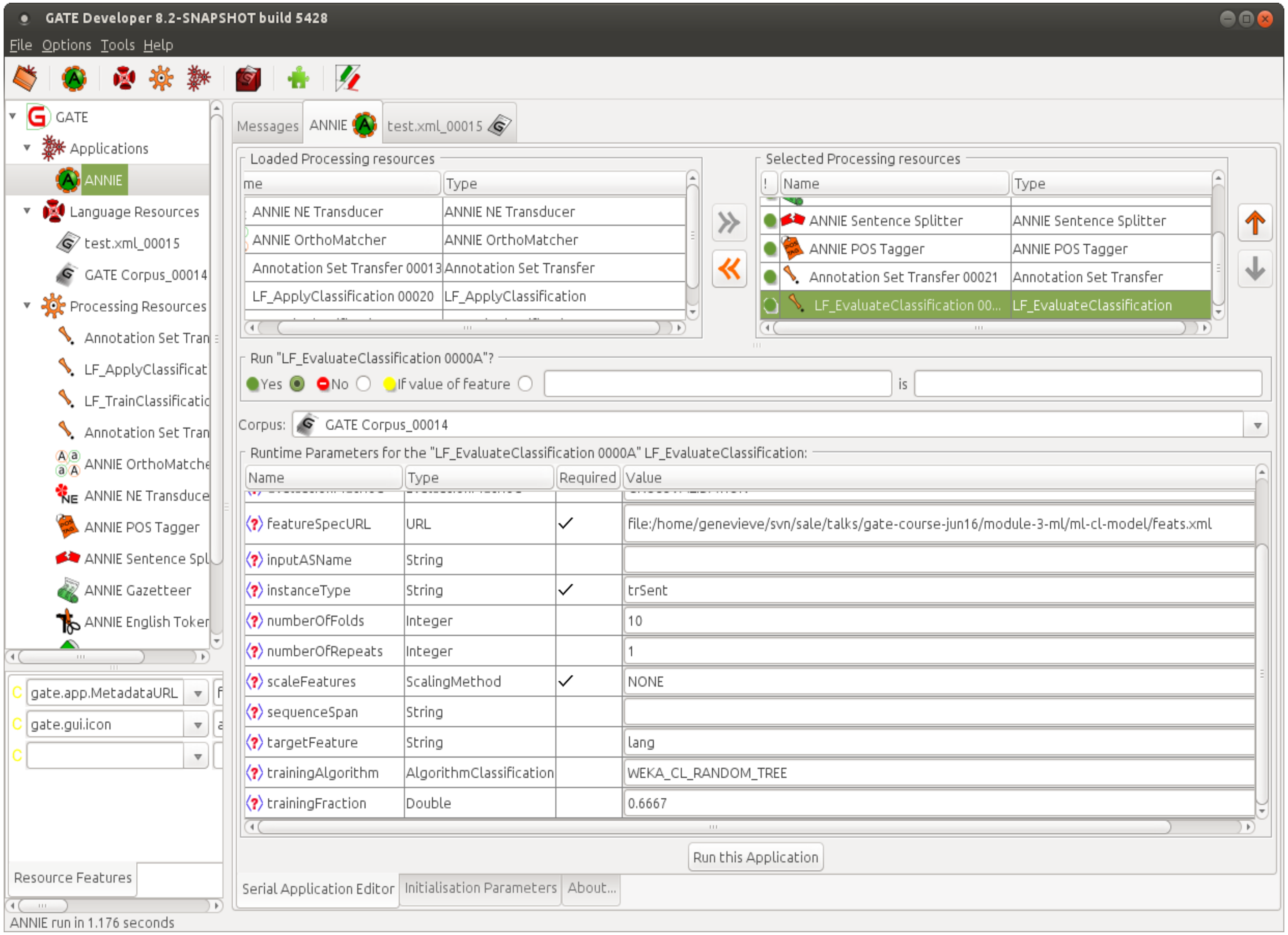
- We have seen that our classifier is not performing as well as we might hope!
- The model has not learned to identify French sentences, and seems biased toward classifying as German
- Maybe we can improve this
- It would be easier to try different things using holdout or cross validation approaches, which would automate the process of splitting, training and testing

Classification using the Evaluation PR

Classification Evaluation PR

- This implements holdout and n-fold cross validation evaluation
- It will split, train and test, and give you an accuracy figure
- It does not create persistent annotations on the corpus that can be examined
- It does not provide a kappa statistic
- However it is a fast way to tune parameters
- We can later return to separate training and application, once we have improved our parameters

Making the Application



- Create and add a classification evaluation PR
- We'll need the annotation set transfer too!

Evaluation PR—Parameters

- We have already introduced some of the parameters, but this PR has several new ones
- classAnnotationType—the annotation type to use as target for chunking*. **Leave blank to indicate classification**
- evaluationMethod—Cross-validation or hold-out
- featureSpecURL—As previously, the xml file containing the feature specification
- inputASName—Input annotation set containing attributes/class (we have everything in the default annotation set)
- instanceType—annotation type to use as instance (whatever you set your AST to create)

*Why would you evaluate chunking using the classification evaluation PR? I'll tell you later!

Evaluation PR—Parameters

- numberOfFolds—number of folds for cross-validation
- numberOfRepeats—number of repeats for hold-out
- targetFeature—for classification only, which feature on the instance annotation (not classAnnotationType!) indicates the class? **Leave blank to indicate chunking**
- trainingAlgorithm—which algorithm to use
- trainingFraction—for hold-out evaluation, what fraction to train on?

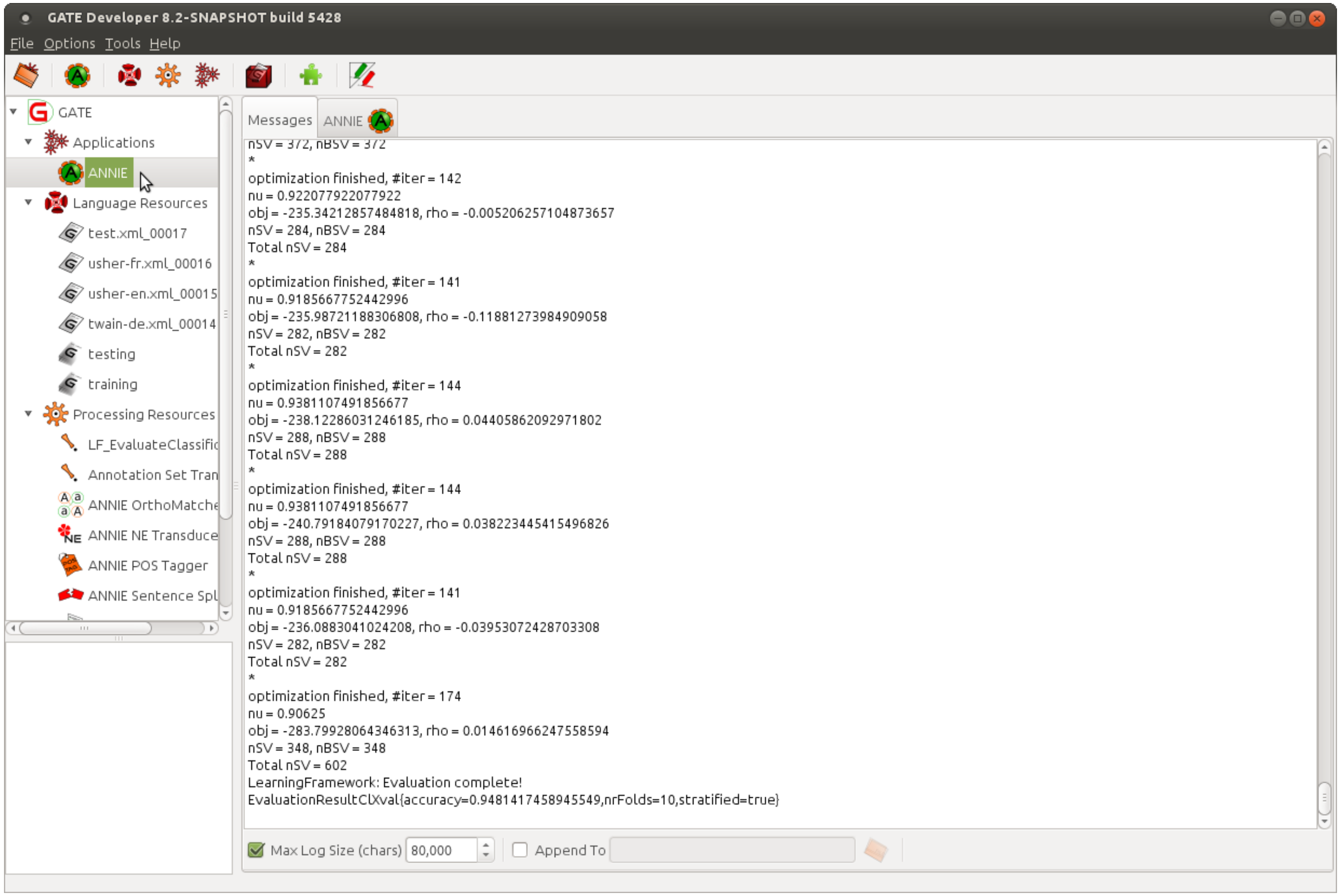
More operations—Evaluation

- Two evaluation modes are provided; CROSSVALIDATION and HOLDOUT
- These wrap the evaluation implementation provided by the machine learning library for that algorithm

Setting the parameters

- Now set the parameters of the evaluation PR
- classAnnotationType MUST be left blank, to indicate that we are running a classification problem
- featureSpecURL should point to the feature file
- instanceType is the annotation type we created when we copied our training sentences over from the Key set
- The more folds you use, the better your result will be, because your training portion is larger, but it will take longer to run—10 is common
- targetFeature is the feature containing the class we want to learn—what will that be?
- Let's try the LibSVM algorithm!

Running the application



- Now run the PR
- If you switch to the messages pane, before running the application by right clicking on the application in the resources pane, you can see the output as it appears

Classification Exercises

- Now see if you can improve your result
- Ideas:
    - Try different algorithms
    - For SVM, it's important to tune cost. Cost is the penalty attached to misclassification. A high cost could result in an overfitted model (it just memorised the training data and may be unable to generalize) but a low cost might mean that it didn't really try to learn! In “algorithmParameters” you can set a different cost like this: “-c 2”. The default cost is 1.
    - Add new features
- Where to get help: https://github.com/GateNLP/gateplugin-LearningFramework/wiki
    - E.g. the Algorithm Parameters page

Chunking—Practical Exercise

Chunking for NER

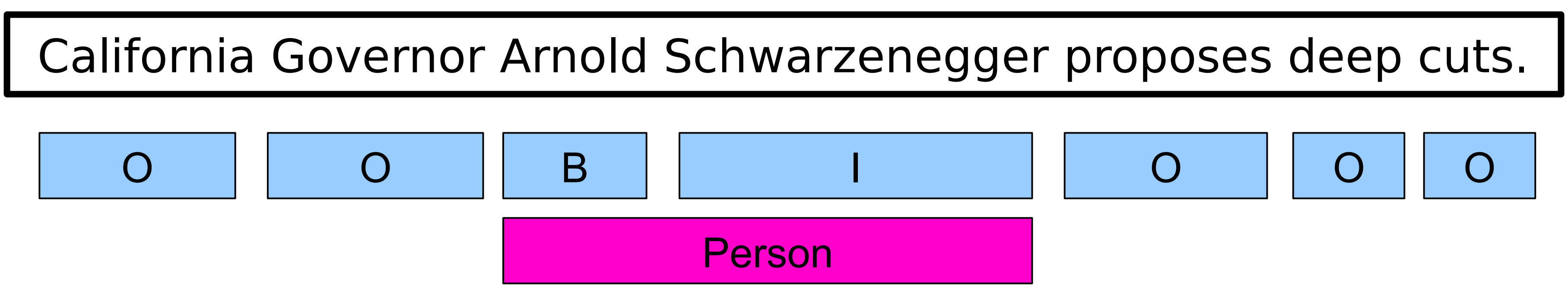
- Chunking, as we saw at the beginning, means finding parts of text
- This task is often called Named Entity Recognition (NER), in the context of finding person and organization names
- The same principle can be applied to any task that involves finding where things are located in text
  - For example, finding the noun phrases
  - Can you think of any others?

California Governor Arnold Schwarzenegger proposes deep cuts.

Person

Chunking for NER

- It's implemented as a twist on classification (everything is classification under the hood!)
- We achieve this in the Learning Framework by identifying which tokens are the beginning of a mention, which are the insides and which are the outsides ("BIO")
  - There are other schemes; the old Batch Learning PR used BE (beginnings and ends)
- You don't need to worry about the Bs, Is and Os; the Learning Framework will take care of all that for you! You just need a corpus annotated with entities



Chunking—Practical Exercise

- Materials for this exercise are in the folder called “chunking-hands-on”
- You might want to start by closing any applications and corpora from the previous exercise, so we have a fresh start

Finding Person Mentions using Chunking Training and Application PRs

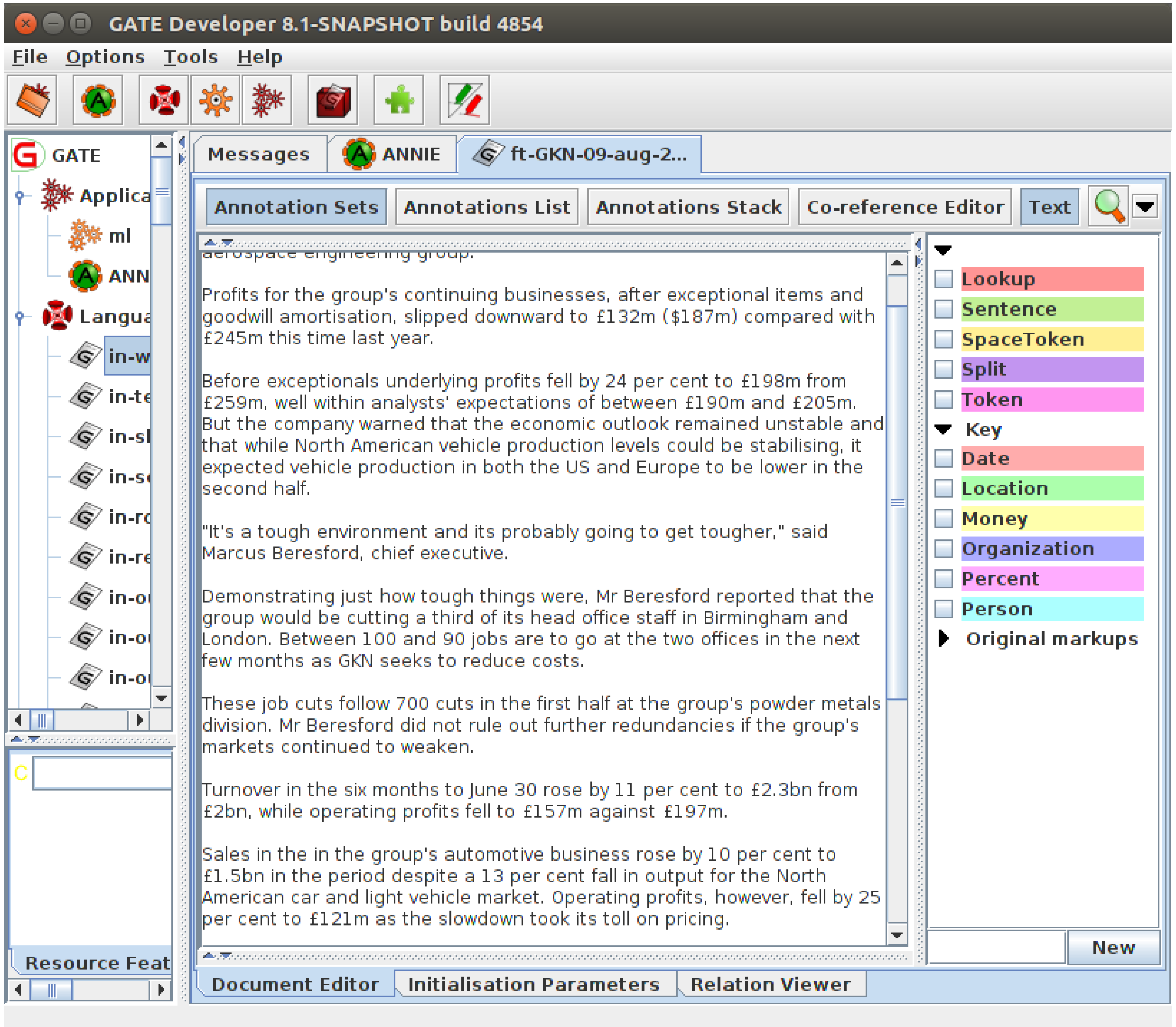
Load the corpus

- Create corpora for training and testing, with sensible names
- Populate them from the training and testing corpora you have in your chunking hands on materials
- Open a document and examine its annotations

Examining the corpus

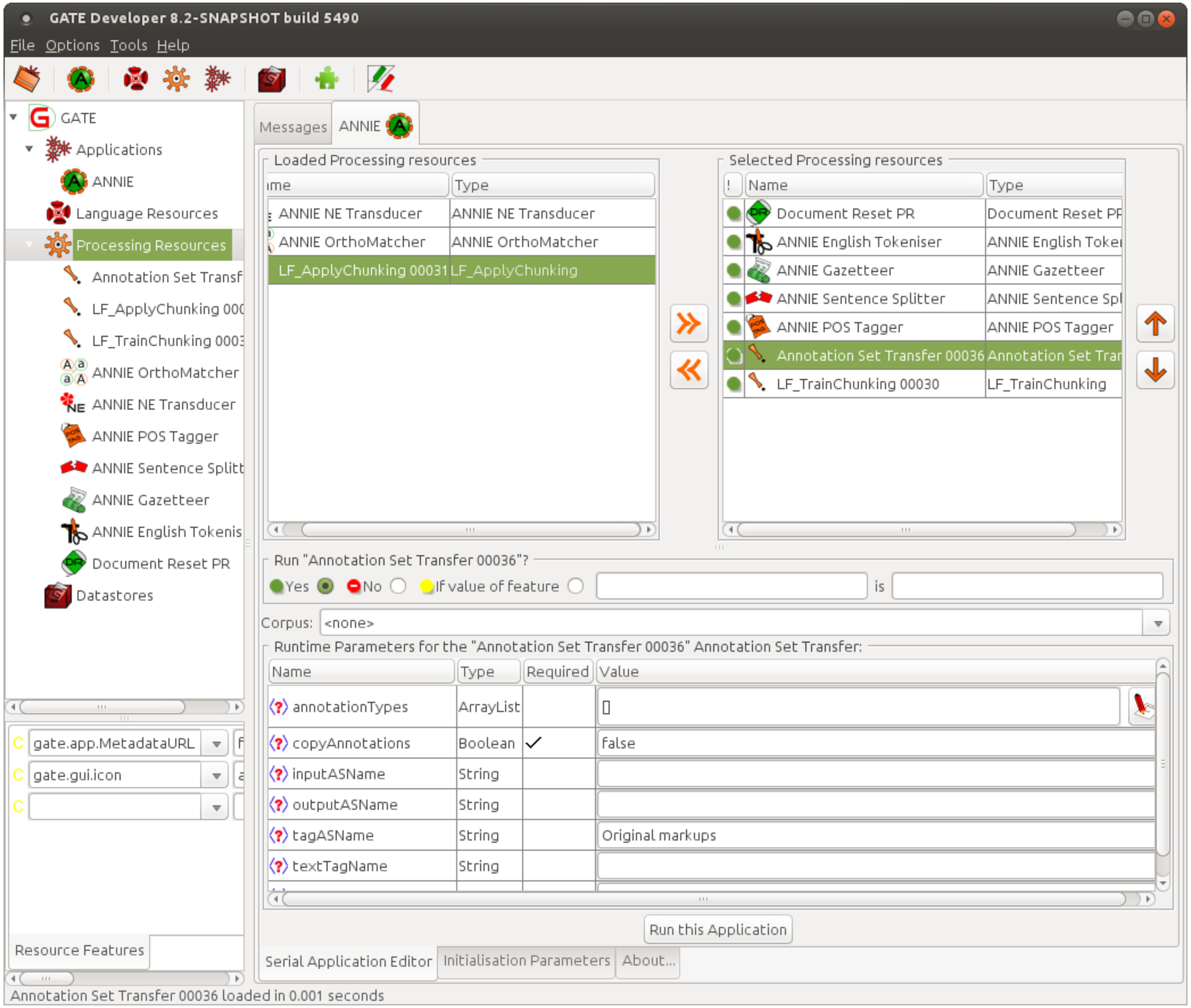
- The corpus contains an annotation set called “Key”, which has been manually prepared
- Within this annotation set are annotations of types “Date”, “Location”, “Money”, “Organization” and so forth

Creating the application



- As previously, if we run ANNIE on the corpus, we have more annotations to work with
- So start by loading ANNIE as the basis for your application
- Again, we don't need the NE transducer or orthomatcher

NER GATE application

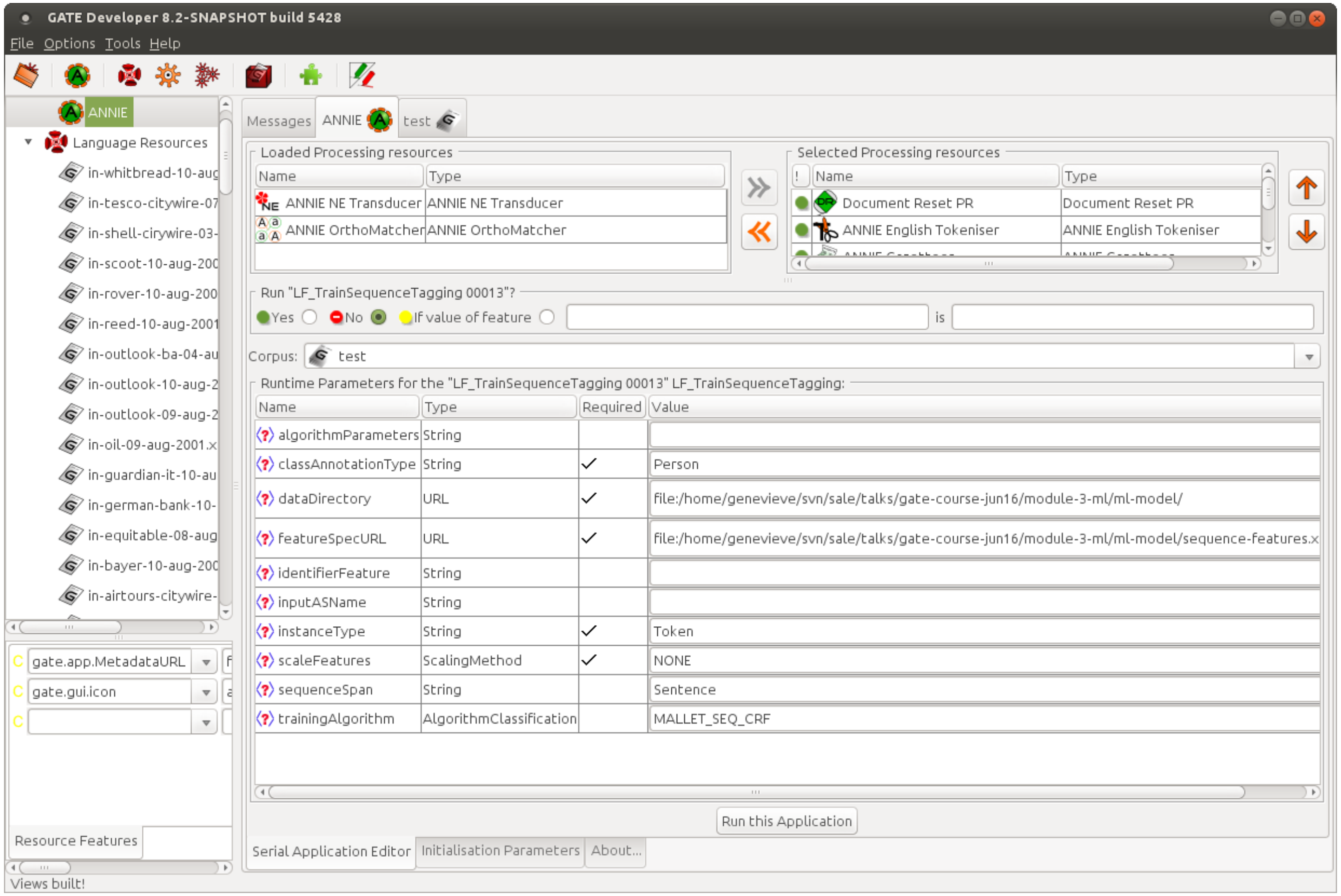


- Again, we need an Annotation Set Transfer, so create and add one
- Then create both training and application chunking PRs
- Start by just adding the training one

Annotation Set Transfer

- We'll use the annotation set transfer to copy the Person annotations up to the default annotation set, where we can learn them
- Go ahead and set up your AST now
- Be sure to copy them, not move them!

Chunking training parameters



- Let's look at the parameters for the training PR
- Instead of targetFeature, we have classAnnotationType

Chunking training parameters

- For classification, the class to learn is in a feature on the instance, is specified to the PR in the targetFeature parameter
- For chunking, the class to learn takes the form of an annotation type. In our case, our corpus is annotated with Person annotations that we are going to learn to locate
- This type to learn is indicated in the classAnnotationType parameter

Chunking training parameters

- Set the classAnnotationType now
- Set the dataDirectory to where you want to save your model, and set the featureSpecURL (there's a feature spec to get you started in the hands on materials)
- Set instanceType. What do you think it should be?

Sequence Spans

- sequenceSpan is only relevant when using sequence learners
- Sequence learners classify each instance in the span by making use of the others
- For example, a noun phrase might be more likely to follow a determiner than a preposition, or a person name might be more likely to follow the word “Mrs”
- The Learning Framework offers the Conditional Random Fields sequence learner
- It might be good for finding Persons, so let's use it!
  - You don't have to use a sequence learner for chunking though
- What do you think would be a good sequence span?

Sequence Spans

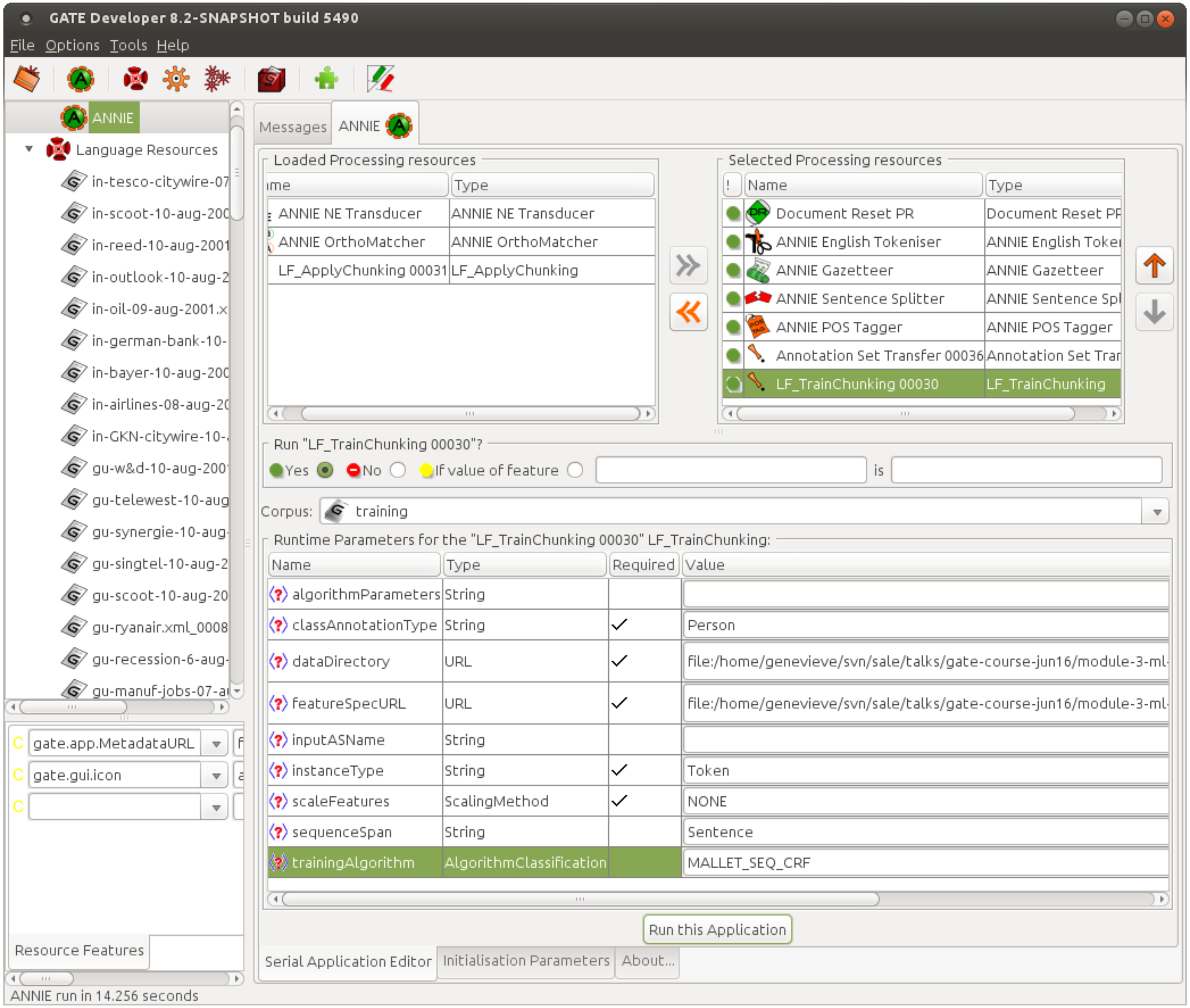
- Sequence spans should be spans within which instance classes follow patterns
  - For example, grammatical rules apply to sequences of parts of speech
  - However, sentiment classifications of individual customer reviews don't form a meaningful sequence
- A sequence span shouldn't be longer than necessary
- Sentence would be a good span for our task
- Fortunately, ANNIE creates sentence annotations for us, so those are available to use
- Set sequenceSpan to “Sentence”

Feature Specification

```
<ML-CONFIG>

<ATTRIBUTE>
<TYPE>Token</TYPE>
<FEATURE>category</FEATURE>
<DATATYPE>nominal</DATATYPE>
</ATTRIBUTE>

<ATTRIBUTE>
<TYPE>Token</TYPE>
<FEATURE>kind</FEATURE>
<DATATYPE>nominal</DATATYPE>
</ATTRIBUTE>

<ATTRIBUTE>
<TYPE>Token</TYPE>
<FEATURE>length</FEATURE>
<DATATYPE>numeric</DATATYPE>
</ATTRIBUTE>

<ATTRIBUTE>
<TYPE>Token</TYPE>
<FEATURE>orth</FEATURE>
<DATATYPE>nominal</DATATYPE>
</ATTRIBUTE>

<ATTRIBUTE>
<TYPE>Token</TYPE>
<FEATURE>string</FEATURE>
<DATATYPE>nominal</DATATYPE>
</ATTRIBUTE>

</ML-CONFIG>
```

- For this task, we are using attribute features
- These allow us to take features from the instance annotations or others that are co-located with them
- We specify type, feature and datatype
- Attribute features also can be taken from instances nearby
- That's a bit less useful with a sequence learner though—why?

Training

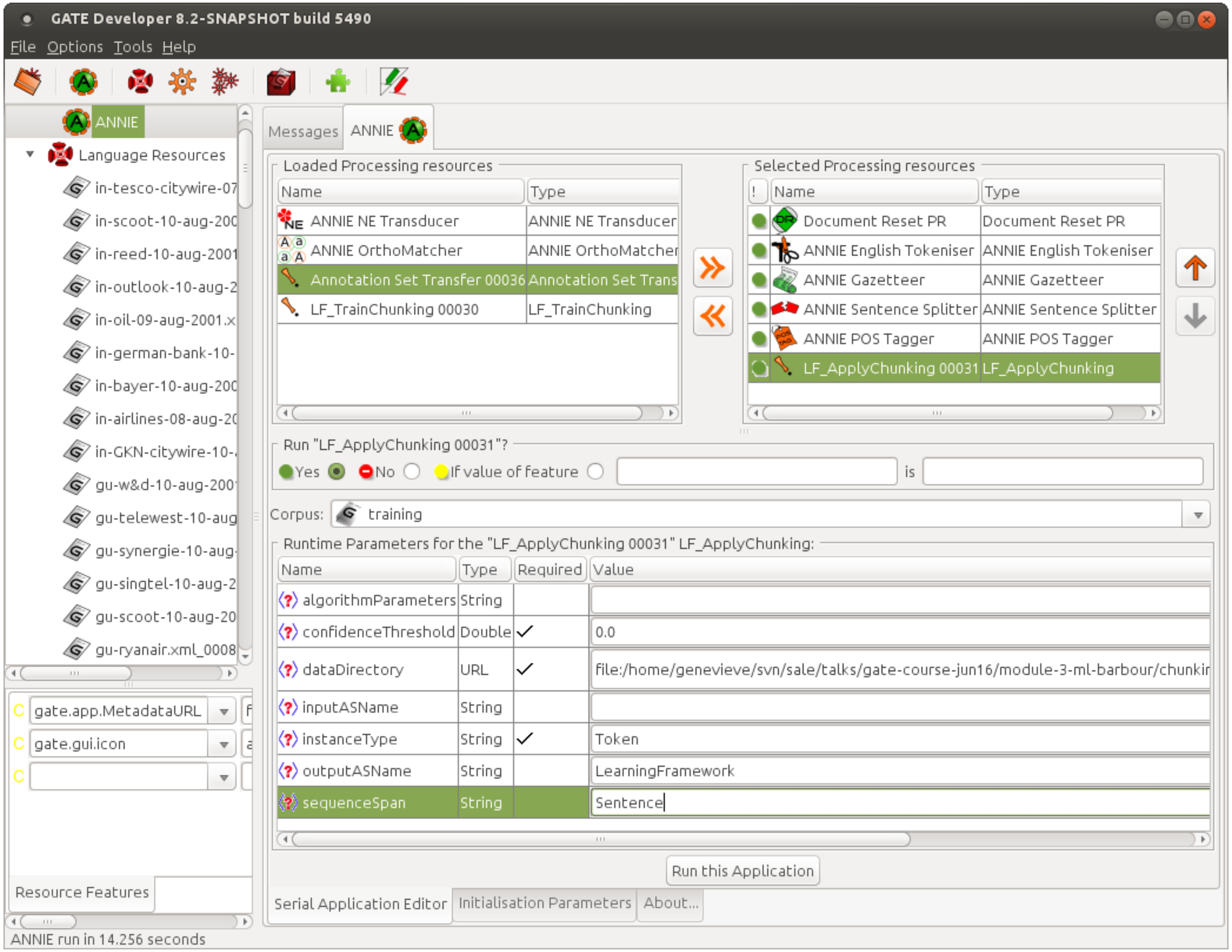


- Make sure you have selected the training corpus
- Run the application!

Chunking application parameters

- Now move the training PR out of the application and add the application PR
- You can take the annotation set transfer out too
- It doesn't have a targetFeature parameter like the classification application PR did
- You don't need to tell it what type to create because the model knows it from training!
- Set dataDirectory to the location where you told the training PR to put the model
- Set the sequence span

Applying



- Now run this on the test corpus

Chunking—Evaluation using Corpus QA

Chunking Evaluation

- For classification, each response is simply right or wrong
- For NER, there are more ways to be wrong
  - Fewer or more mentions than there really are, or you can overlap
- So we need different metrics

What are precision, recall and F1?

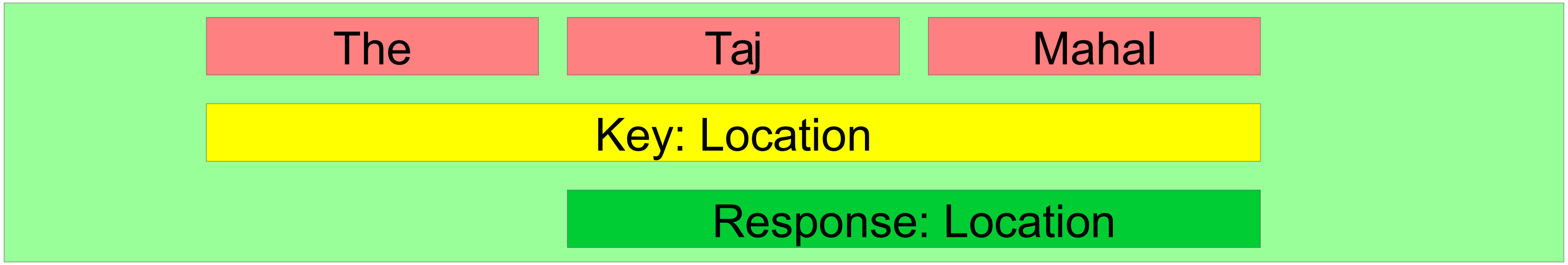
- Precision: what proportion of our automatic annotations were correct?
- Recall: what proportion of the correct annotations did our automatic tool create?
- P = correct / (correct + spurious) = tp / (tp + fp)
- R = correct / (correct + missing) = tp / (tp + fn)
- where tp = true positives, fp = false positives, fn = false negatives

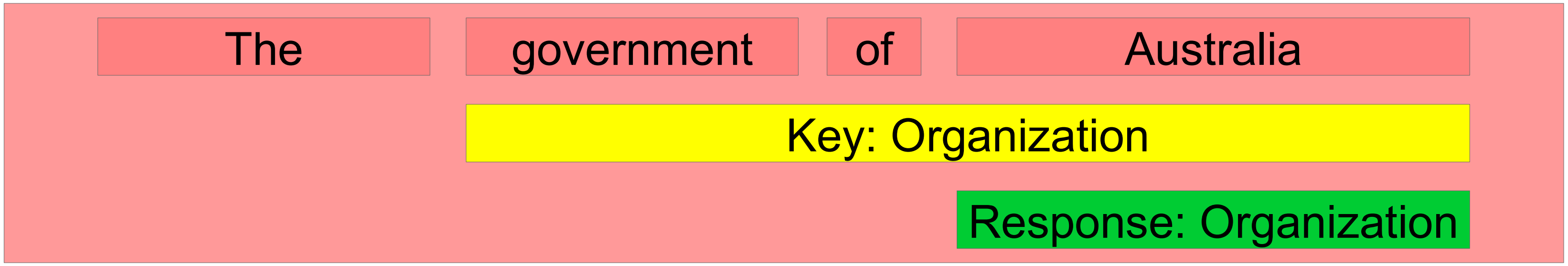
What are precision, recall and F1?

- F-score is an amalgam of the two measures
  - $F_\beta = (1+\beta^2)PR / (\beta^2 P + R)$
    - The equally balanced F1 ($\beta = 1$) is the most common F-measure
    - $F1 = 2PR / (P + R)$

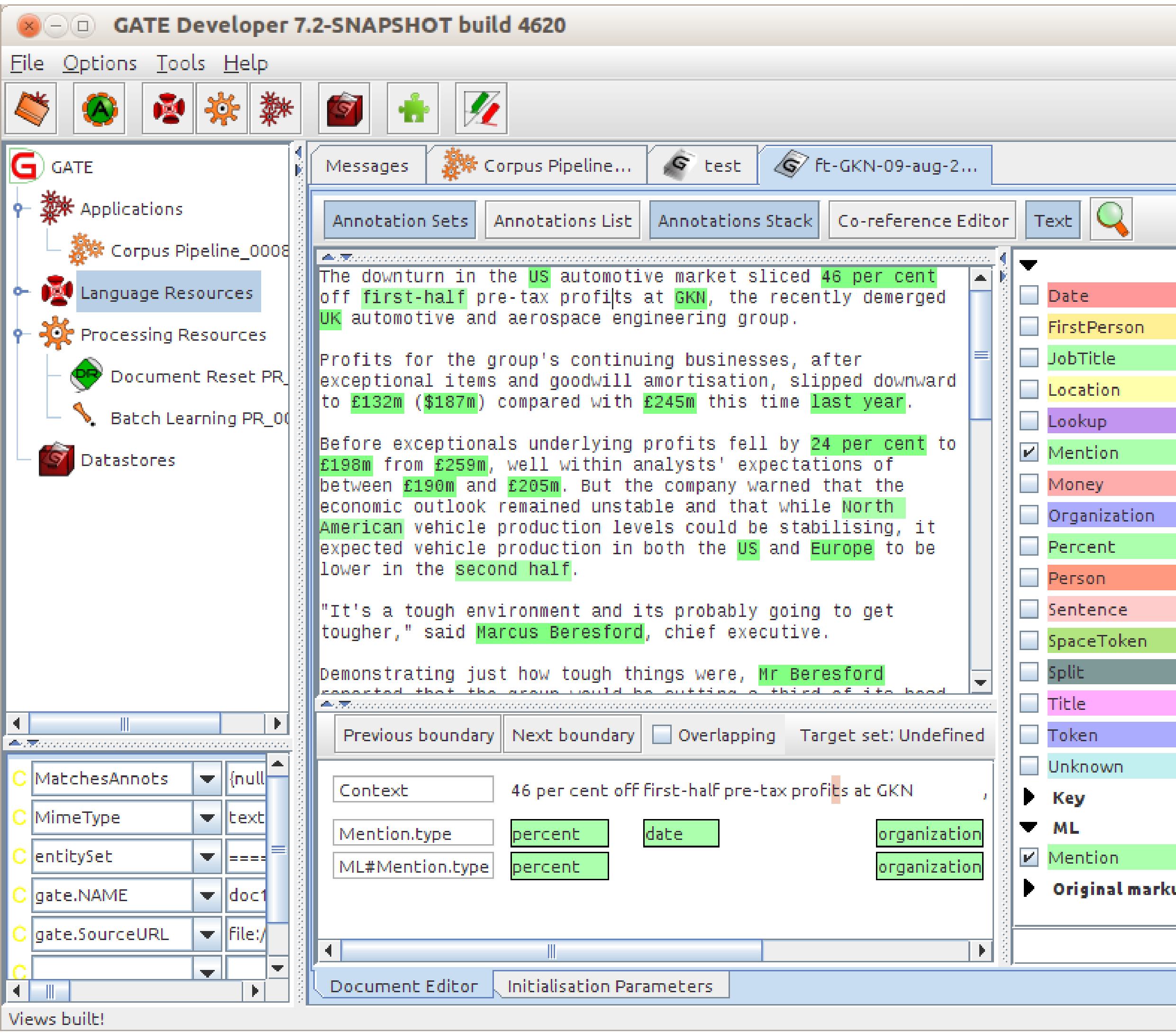
Strict and Lenient

- “Strict” means we count an annotation as correct only if it has the same span as the gold standard annotation
- Lenient means we allow an annotation that overlaps to be correct, even if it isn't a perfect span match
- Which do you think is the right way to do it?



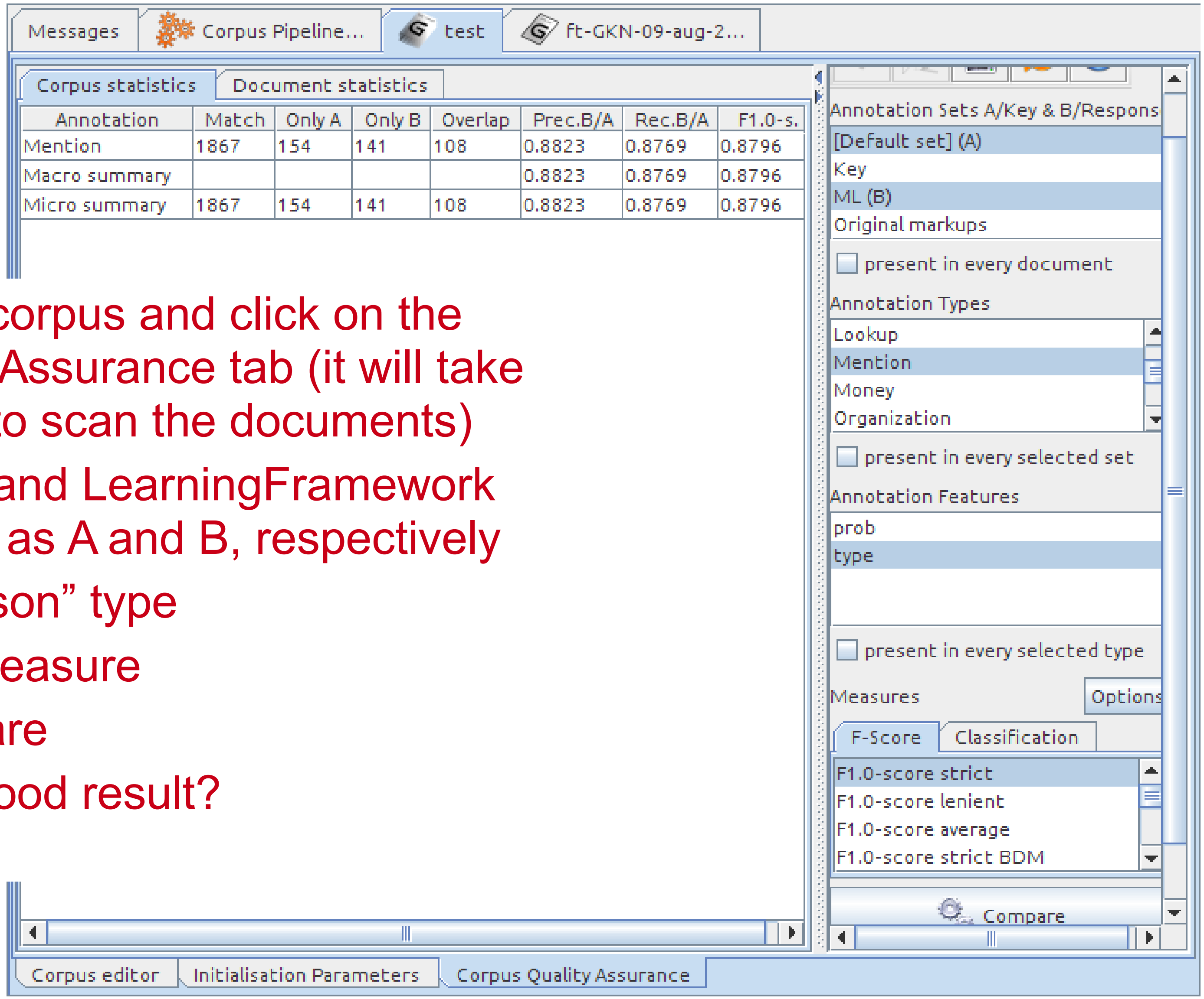


Examining the results of application



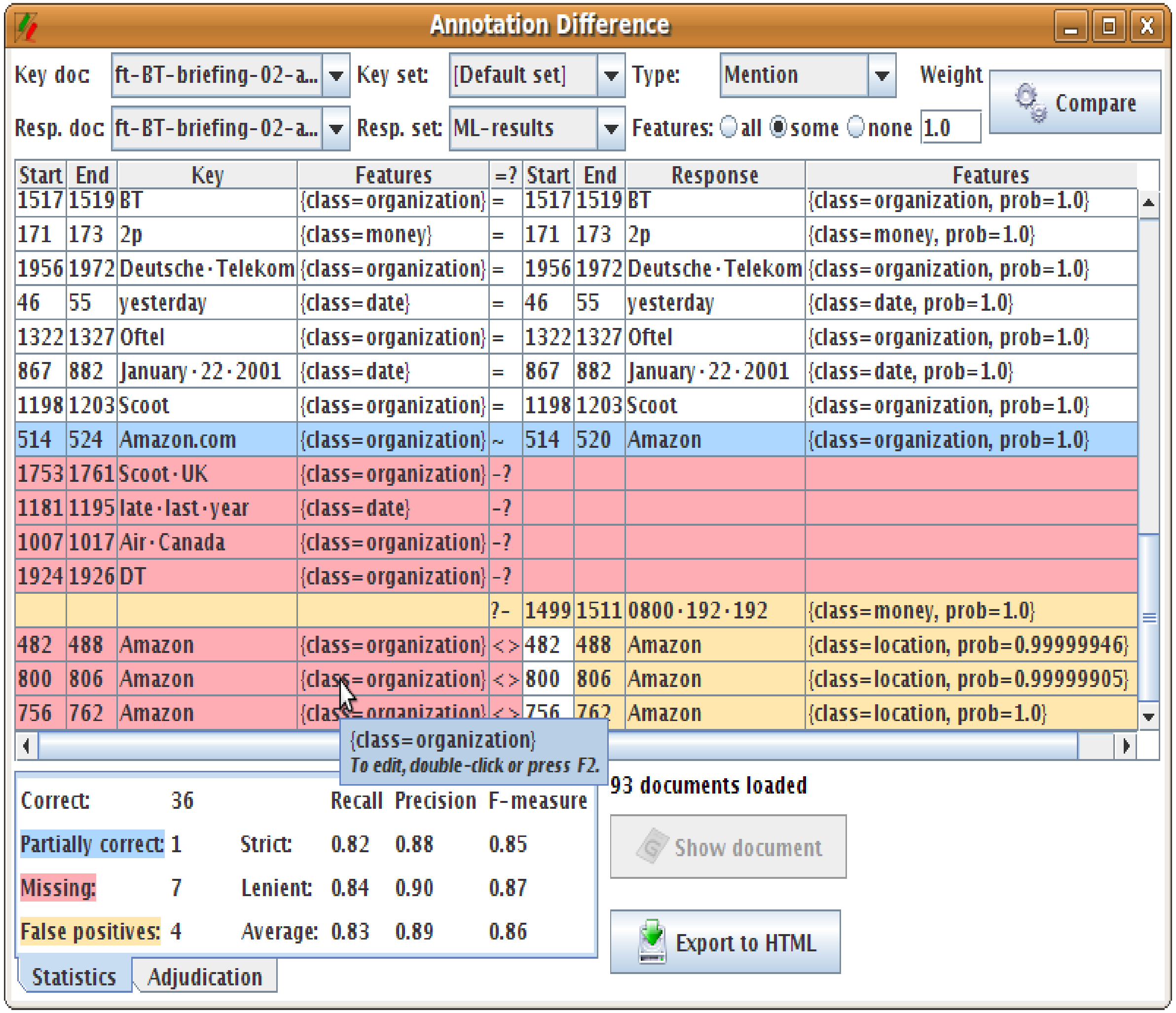
- **Examine a document from the test corpus**
- You should have a new “LearningFramework” AS with Person annotations
- The original Person annotations (in the Key AS) are similar but not always identical!
- The Annotations Stack is good for comparing them
- How similar do they appear to be? Do you think you will get a good result?

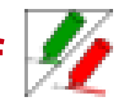
Comparing the Sets with Corpus QA



- Select the test corpus and click on the Corpus Quality Assurance tab (it will take a few seconds to scan the documents)
- Select the Key and LearningFramework annotation sets as A and B, respectively
- Select the "Person" type
- Choose an F-measure
- Click on Compare
- Did you get a good result?

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?

Using Annotation Diff...

- “Correct”: the response annotation has the right feature and span
- “Partially correct”: response has the right feature and overlapping but not exactly matched span; this counts as correct in the “lenient” scoring
- “Missing”: key annotation+feature is missing from the response (a.k.a. “false negative”)
- “False positive”: response annotation+feature shouldn't be there (a.k.a. “spurious”)

Classification Evaluation PR for Chunking?

- We didn't use a Learning Framework evaluation PR for this chunking task
- What do you think would happen if you used the Classification Evaluation PR to do a chunking problem?
- It would work! It would evaluate the accuracy of the system in correctly identifying beginnings, insides and outsides
- However, it wouldn't tell you much about how well you did finding named entities
  - There are so many outsides that you can get a high score just by saying everything is an outside!
- You could use it to tune parameters if you wanted, though

Exercise—Improving the result

- Again, see if you can improve your result
- Try different features and algorithms

Exercise 2

- Try to learn different entity types

Exporting Feature Data

Exporting feature data

- A GATE ML PR serves a number of functions
  - Scraping features off the documents and formulating them as ML training sets
  - Sending the training sets to ML libraries to train a model
  - Creating instances (without class) at apply time to send to a trained model to be classified and writing the resulting class back onto the application instance
- We have integrated quite a few algorithms and some ML facilitation technology, so many ML tasks can be accomplished entirely in GATE

Exporting feature data

- However, GATE isn't an ML tool—its forte and contribution is complex linguistic features. There is a limit to what we will include in the way of ML innovations.
- For example, the Learning Framework;
  - doesn't include feature selection technologies
  - includes only limited feature scaling
  - doesn't integrate all algorithm variants

Exporting feature data

- For more advanced needs, there are other ways to work
- You can export your training set and use it to train a model outside of GATE
  - The Learning Framework will allow you to use a model trained outside of GATE to create an application
- Exporting data and working in e.g. Weka can also provide a faster way to tune parameters
  - When you change parameters in the LF it starts over again scraping the features off the documents, which takes a long time on a big corpus
- You could use e.g. Weka's feature selection technology and bring what you learned back into GATE by editing your feature spec
- It can also be a good sanity check to see your data in export format

Export the data as ARFF

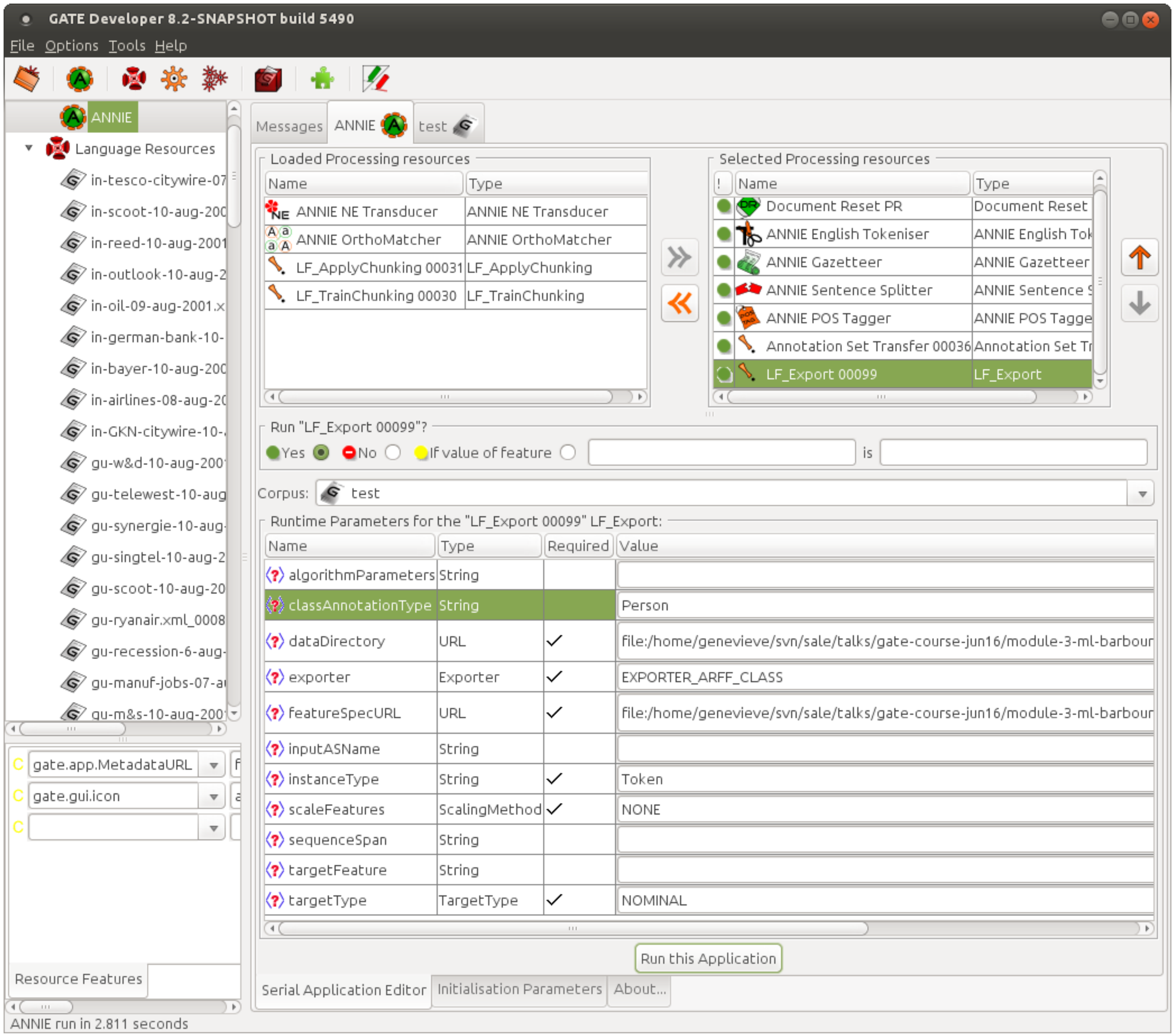
- Create an Export PR and add it to the application
- You can remove the other Learning Framework PRs
- Annotation Set Transfer needs to stay though

Export Parameters

- classAnnotationType is as for training, and its presence indicates that we are exporting a CHUNKING dataset—set it to Person
- dataDirectory, featureSpecURL, inputASName and instanceType you are familiar with by now—set them
- For exporter, choose EXPORTER_ARFF_CLASS*
- Don't set target feature! This would indicate that we want to export a classification dataset!
- Don't set sequenceSpan—this would indicate that we want to export data in a format suitable for training a sequence learner. This isn't supported yet.

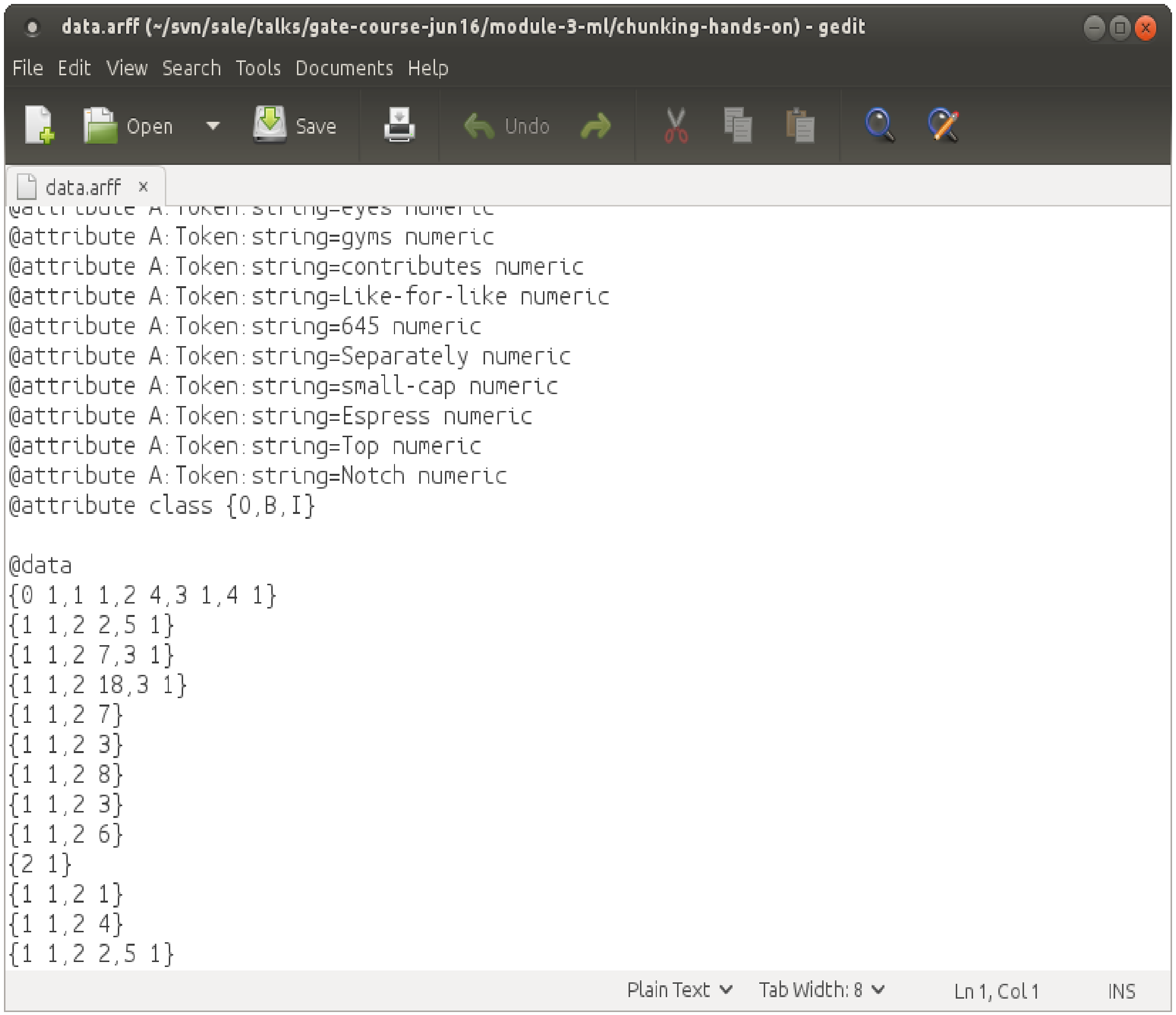
* ”CLASS” means classification—why are we exporting a classification dataset for a chunking problem? Because they're all classification behind the scenes. GATE turns the chunking problem into a classification problem for training and then turns it back again!

Exporting



- Set targetType to nominal, because beginnings, insides and outsides are nominal classes
- Go ahead and export the data!

Examining the ARFF




```
@attribute A:Token:string=gyms numeric
@attribute A:Token:string=contributes numeric
@attribute A:Token:string=Like-for-like numeric
@attribute A:Token:string=645 numeric
@attribute A:Token:string=Separately numeric
@attribute A:Token:string=small-cap numeric
@attribute A:Token:string=Espress numeric
@attribute A:Token:string=Top numeric
@attribute A:Token:string=Notch numeric
@attribute class {O,B,I}

@data
{0 1,1 1,2 4,3 1,4 1}
{1 1,2 2,5 1}
{1 1,2 7,3 1}
{1 1,2 18,3 1}
{1 1,2 7}
{1 1,2 3}
{1 1,2 8}
{1 1,2 3}
{1 1,2 6}
{2 1}
{1 1,2 1}
{1 1,2 4}
{1 1,2 2,5 1}
```

- You'll find your exported ARFF in your dataDirectory, called data.arff
- Examine it now
- At the top are a list of attributes. Are they as expected?
- The last attribute is the class attribute. Do you see it?
- After that come feature vectors in sparse format. How can you tell that they are in sparse format? What would this file look like if they were written out in full?

Working with Weka

Why would I want to use Weka?

- As noted previously, Weka can be faster and better for playing around with parameters to get the best result
  - Now that you have exported your data, you can try loading it into Weka in your own time, and see what you can do there
- But then you need to bring that result back into GATE! So you need to run the Weka algorithm in GATE
- Weka has some good algorithms that might be better for your task
  - Though note that Mallet's CRF is often the best for chunking, and LibSVM is often the best for most things, and you don't need Weka for those
- However, due to licensing incompatibility, we can't integrate Weka into GATE as seamlessly as we integrated LibSVM and Mallet

What you need

- Weka integration comes as a separate project, but it's easy to do!
- You need to get the Weka wrapper from here (downloading the zip is easiest):

  https://github.com/GateNLP/weka-wrapper/

- You need to tell your application where to find the Weka wrapper
  - Use the environment variable WEKA_WRAPPER_HOME
  - Or use the java property gate.plugin.learningframework.wekawrapper.home
  - Or the setting wekawrapper.home in a file weka.yaml in the data directory used

Using Weka in the GATE GUI

- Then you can go ahead and use Weka for classification and chunking by:
  - Creating a training PR
  - Selecting WEKA_CL_WRAPPER for trainingAlgorithm
  - Giving the full class name of the Weka algorithm as the first algorithmParameters argument
    - For example “weka.classifiers.trees.RandomForest”
  - A model will be created in the specified directory as before
  - At apply time, you simply indicate this model as usual
- (Weka in the evaluation PR isn't supported—try using Weka to evaluate!)

Where to find documentation about ...

- Getting the Weka wrapper and using it to train models outside of GATE:
  - https://github.com/GateNLP/weka-wrapper
- Using Weka inside of GATE:
  - https://github.com/GateNLP/gateplugin-LearningFramework/wiki/UsingWeka
- What Weka algorithms' full class names are:
  - Weka's Javadoc, e.g. http://weka.sourceforge.net/doc.dev/weka/classifiers/Classifier.html
- Note that the Weka wrapper is very new code! Let us know if you find any problems with it!