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 features be?
- If we run parts of ANNIE over the corpus, then we can use:
  - Sentence annotations for instances
  - Token features for attributes
- We can also use the feature generation PRs, which require tokens

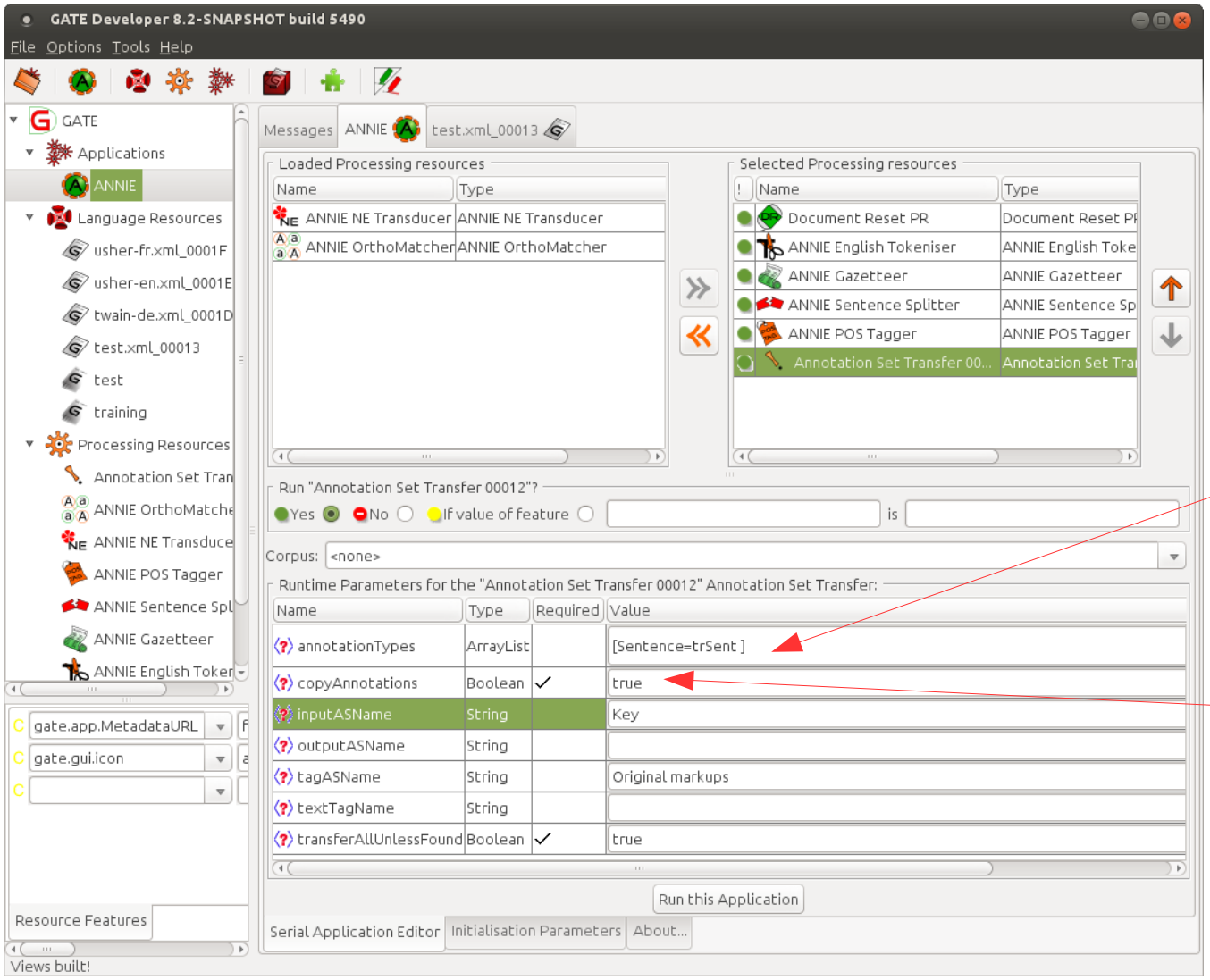
Making the Application

- Load ANNIE
- 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

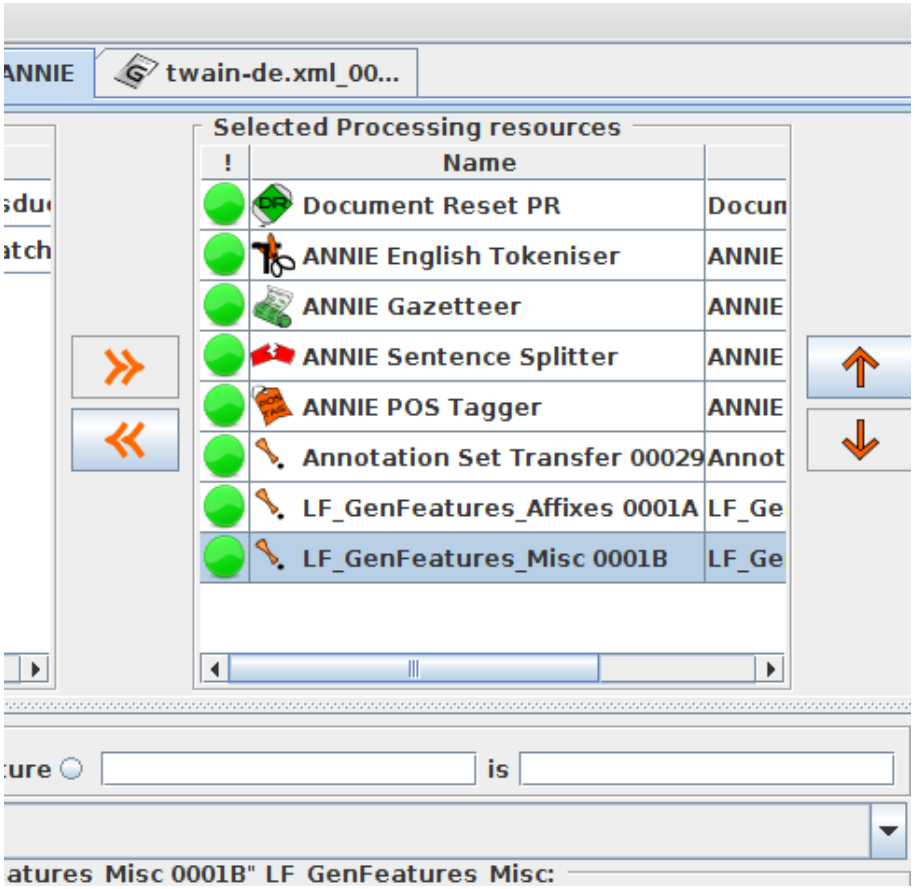
- The Learning Framework expects all class and feature annotations to be in the same set
- ANNIE puts annotations in the default set
- So we need to copy the sentences from Key into the default set
  - (We could have ANNIE output to “Key” but it would be a lot more hassle, and “Key” should be reserved 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”!!!!

Feature Generation PRs



- Create and add LF_GenFeatures_Affixes and LF_GenFeatures_Misc PRs to the application

Affix Feature Generation Parameters

- The PR generates features that are the first or last few characters, because these are often good features. “ing” for example can be good for part of speech tagging
- genPrefixes/genSuffixes – do we want to generate prefixes, suffixes or both?
- inputASName – which annotation set is the annotation in that we want to put features on? We want to put them on “Token”, which ANNIE will generate for us in the default annotation set
- instanceType – the annotation type to put the affixes on
- mapToUpper – do we want to normalize case?

Affix Feature Generation Parameters

- Affixes will be generated for all lengths within the range indicated by maxSuffixLength and minSuffixLength, likewise maxPrefixLength and minPrefixLength. However the maximum affix length is limited by minNonSuffixLength/minNonPrefixLength, which makes sure you don't consider too much of the word to be affix (“sing” for example is unlikely to be a present participle because “s” is an unlikely verb)
- stringFeature indicates the feature on the instance that contains the string
- prefixFeatureName and suffixFeatureName indicate what to call the new features

Misc. Feature Generation Parameters

- "Word shape" provides an abstraction over the orthography of the word
- Every character in the original word string is mapped to one of the following:
  - any upper case letter is mapped to "A"
  - any non-upper case letter is mapped to "a"
  - any numeric digit is mapped to "9"
  - all other characters are copied
- So the wordshape for the word "C6-hydroxylation" is "A9-aaaaaaaaaaaaa"
- For the short word shape feature, the same mapping is performed, but multiple subsequent characters of the same type are all mapped to the same single output characters.
- So the short wordshape for the word "C6-hydroxylation" is "A9-a"
- We'll use the short word shape here

Training PR

- Set the Affixes PR to generate suffixes
- Make a PR for classification training and add it to the application at the end

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>wordShapeShort</FEATURE>
</NGRAM>

<NGRAM>
<NUMBER>1</NUMBER>
<TYPE>Token</TYPE>
<FEATURE>suf2</FEATURE>
</NGRAM>

<NGRAM>
<NUMBER>1</NUMBER>
<TYPE>Token</TYPE>
<FEATURE>suf3</FEATURE>
</NGRAM>

<NGRAM>
<NUMBER>1</NUMBER>
<TYPE>Token</TYPE>
<FEATURE>suf4</FEATURE>
</NGRAM>

</ML-CONFIG>
```

- This file is in your hands-on materials
- Feature specification indicates which features we are going to use
- This one uses the word shape and suffixes of every token in the sentence
- What else might be useful for identifying the language a sentence is written in?

Algorithms

- Two libraries are fully integrated; Mallet (providing many algorithms) and and LibSVM (support vector machine)
- Other libraries require a separate download due to licensing reasons. It is easy to do.
  - Costcla, Keras, Pytorch, Scikit Learn and Weka
- 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!

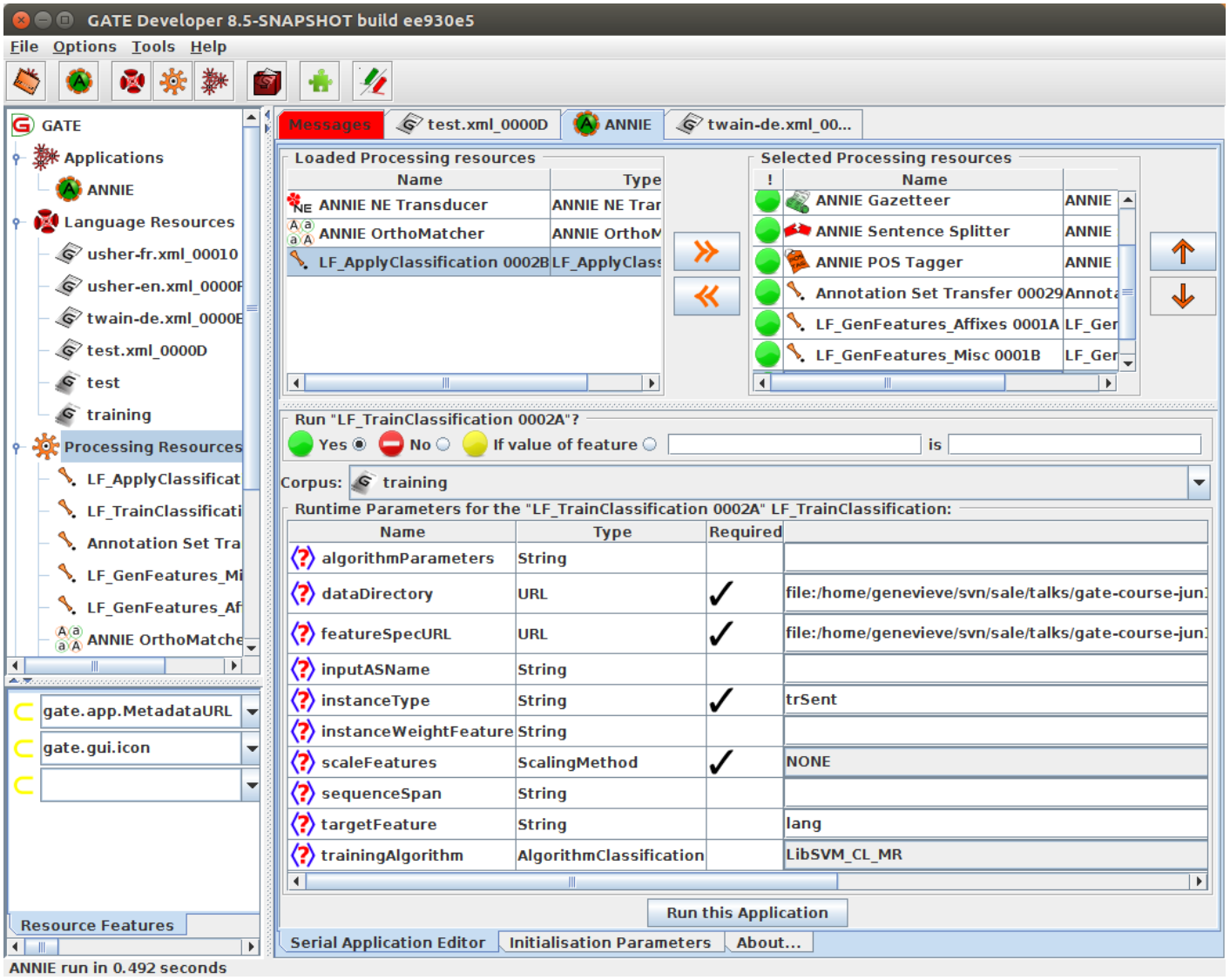
Algorithm Name Codes

- SEQ vs CL
  - SEQ means it's a sequence algorithm (it classifies a whole sequence of instances as a unit)—more about this later
- MR vs DR
  - GATE's main contribution to ML is the feature creation—running the app on the corpus to create data the ML can use
  - After that, it just passes the whole corpus representation to the learning algorithm, and learning begins
  - Different algorithms want different representations
  - “MR” is a corpus in Mallet format, which is sparse (more about this later) and held in RAM
  - “DR” is dense, and is stored on disk

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!

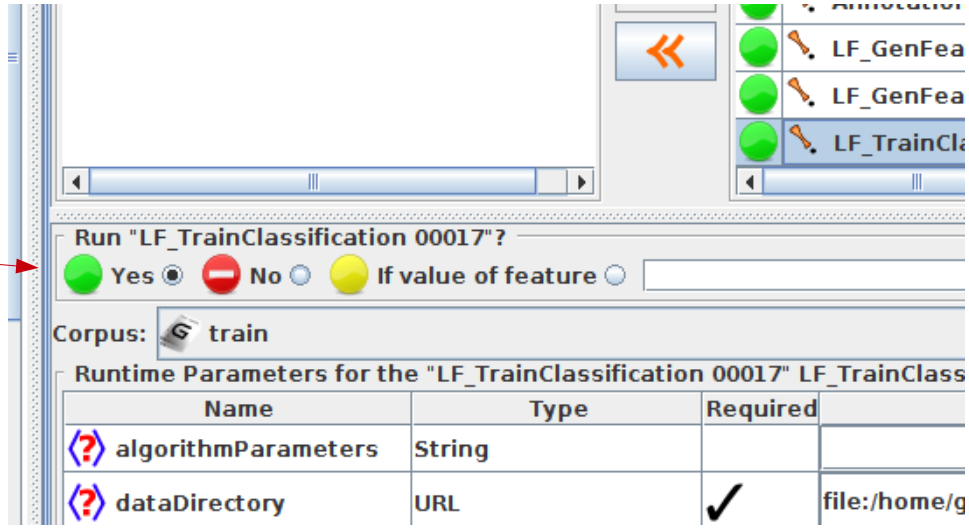
Training a model

- 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

- Switch off the training PR
- You can also switch off 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

Turn off PR with traffic lights



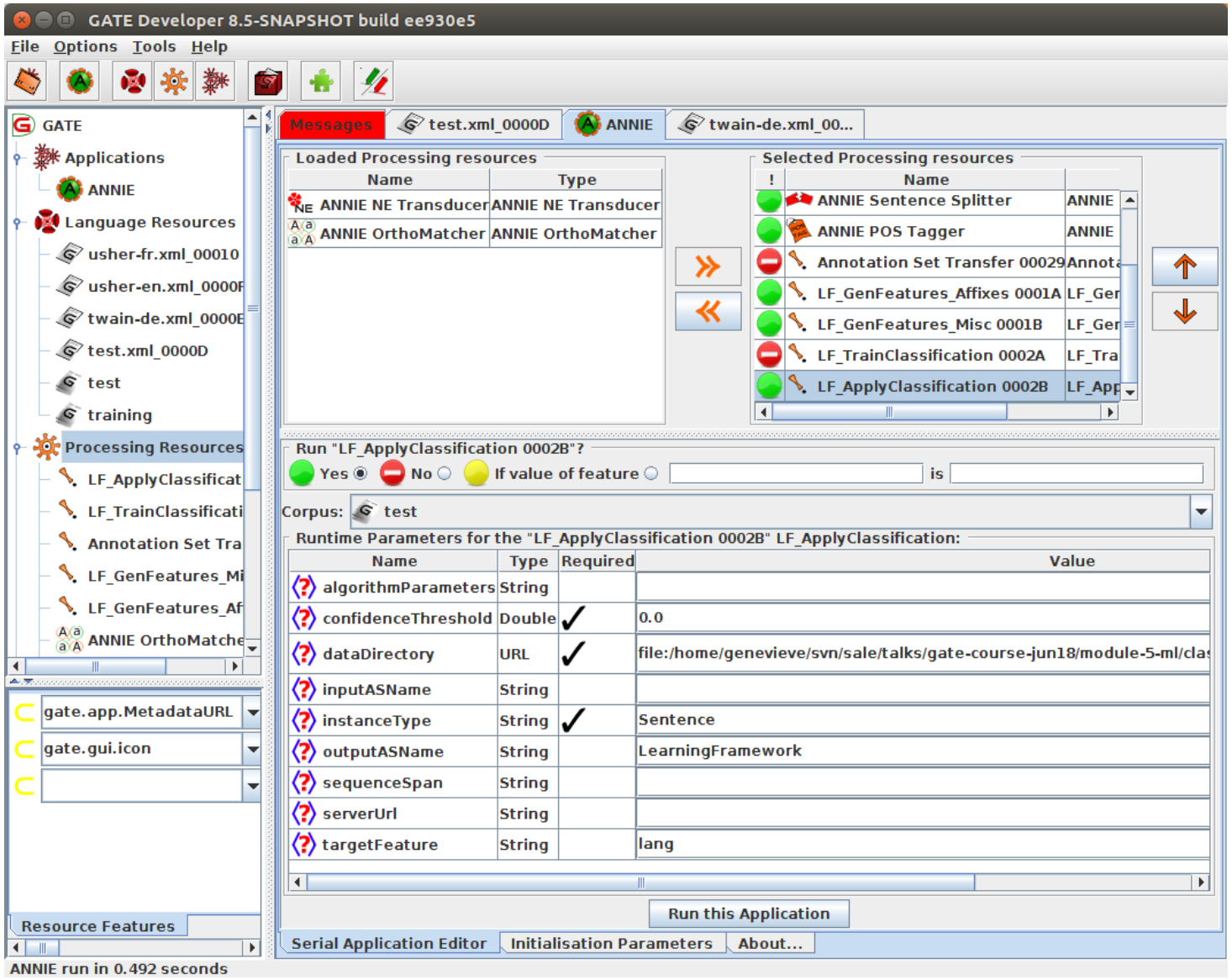
Classification Application

- Create and add an application PR
- 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

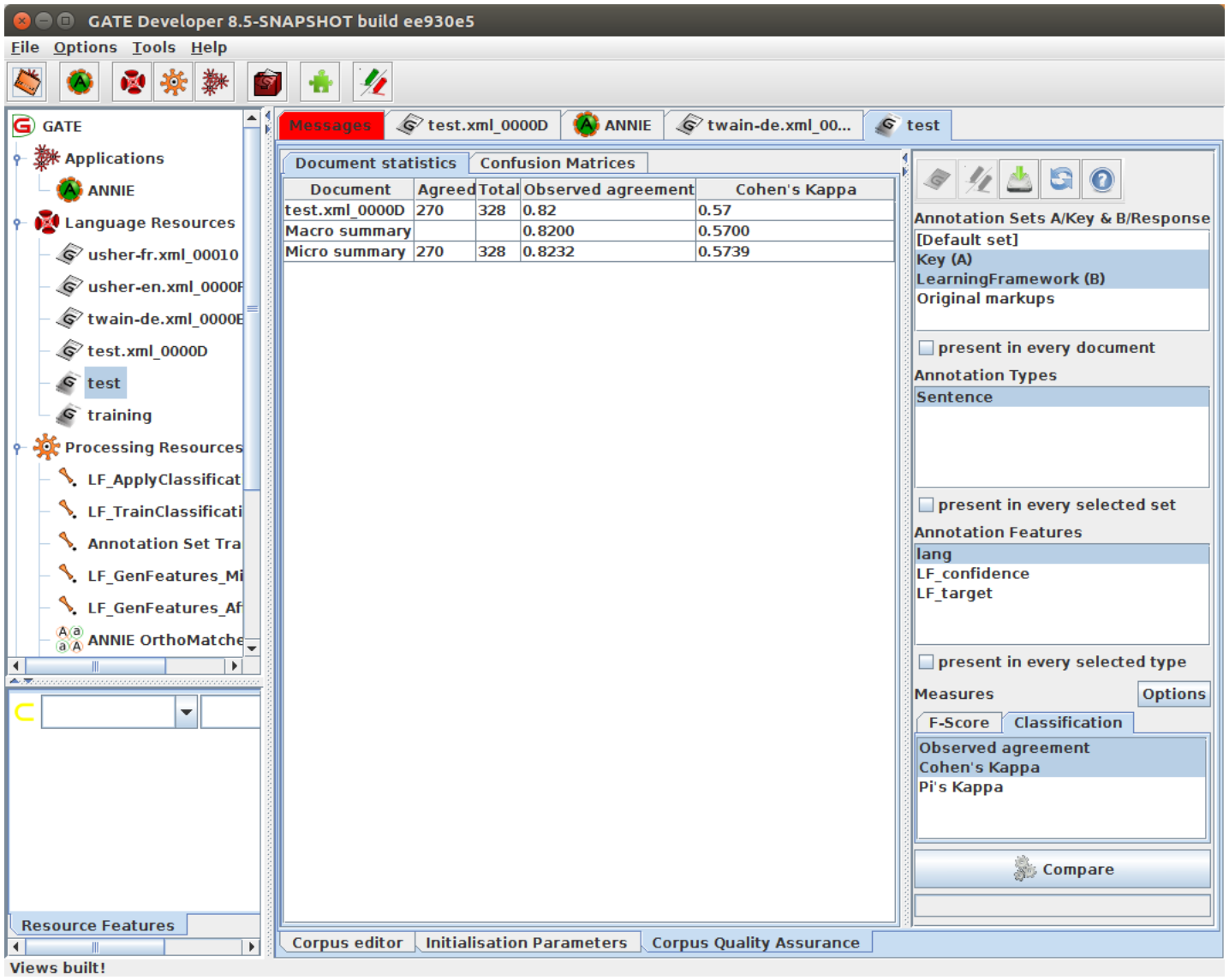
Go ahead and run the application!

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

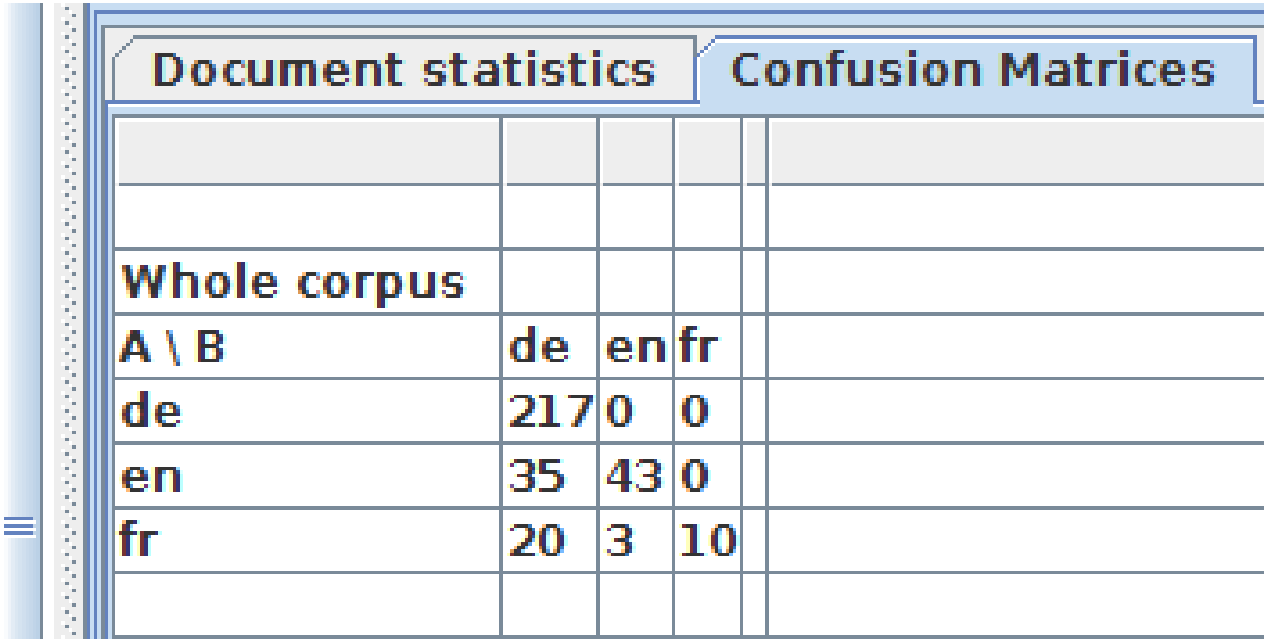
- What do you think about this result? Not bad?
- What do you think of this kappa statistic? (A kappa of over 0.5 is considered good, and over 0.8 excellent.)

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?



Document statistics | Confusion Matrices

Whole corpus

A \ B	de	en	fr
de	217	0	0
en	35	43	0
fr	20	3	10

Classification Evaluation

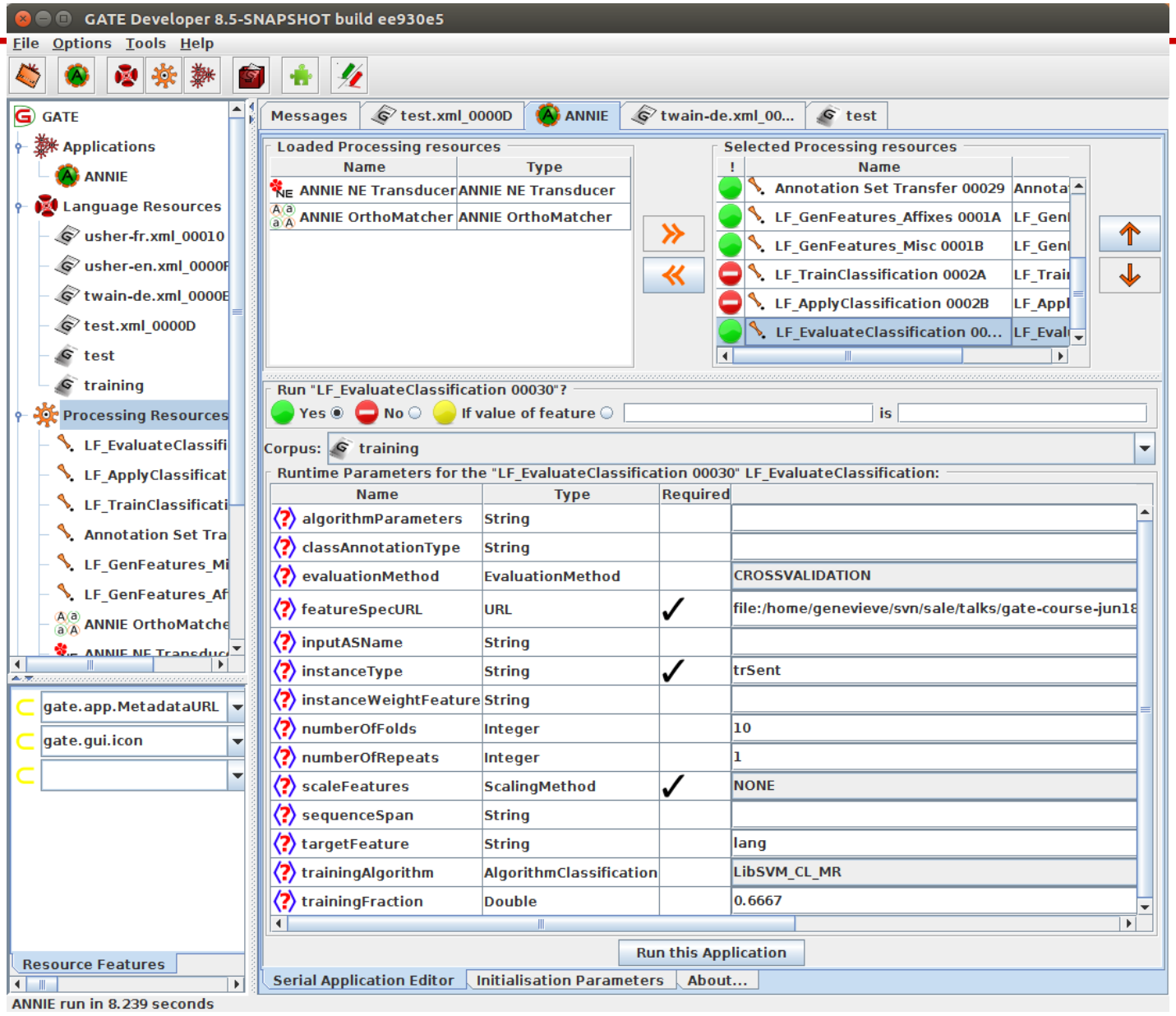
- It is not bad, but it is bad at French, and seems biased toward classifying as German
- Maybe we can improve this
- It would be easier to try different things using holdout or cross validation approaches, which would automate the process of splitting, training and testing

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!
- (But not the application PR)

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? To indicate chunking, you would leave this blank
- trainingAlgorithm—which algorithm to use
- trainingFraction—for hold-out evaluation, what fraction to train on?

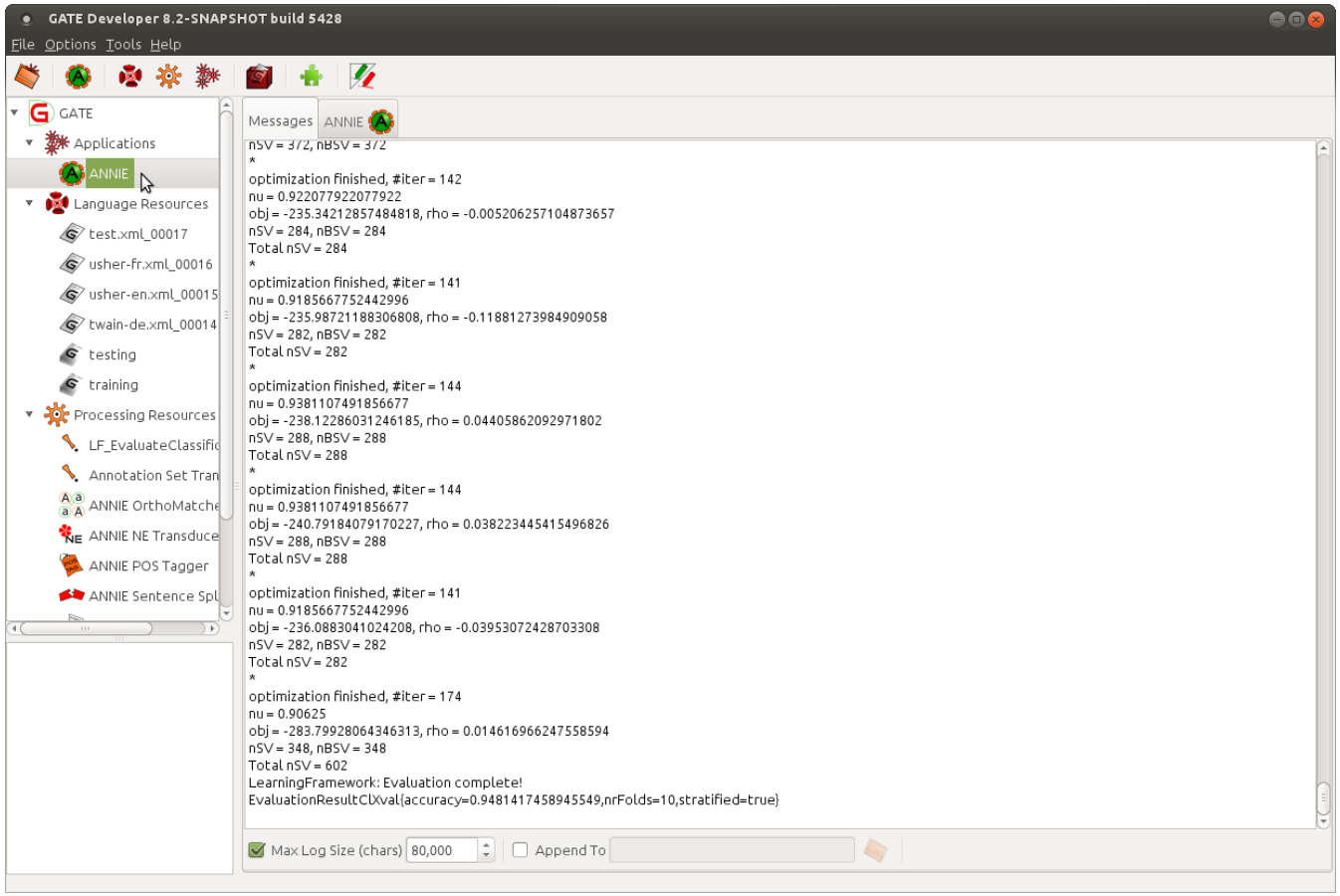
More operations—Evaluation

- 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 (on the training corpus)
- 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
- (The result is a little better than with the separate PR because the training data is more homogeneous)

Classification Exercises

- Now see if you can improve your result
- Ideas:
  - Try different algorithms (not the wrappers or the SEQ ones. **Note that C45 takes longer to run**)
  - 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” or any number you want. The default cost is 1.
  - Add new features
- Where to get help:

  `https://gatenlp.github.io/gateplugin-LearningFramework/`