
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

The screenshot shows the GATE Developer interface with the 'Annotation Set Transfer' tool selected in the 'Processing Resources' list. The 'Runtime Parameters' table is visible below the tool configuration.

Name	Type	Required	Value
annotationTypes	ArrayList		[Sentence=trSent]
copyAnnotations	Boolean	✓	true
inputASName	String		Key
outputASName	String		
tagASName	String		Original markups
textTagName	String		
transferAllUnlessFound	Boolean	✓	true

- 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



The screenshot shows the GATE Developer 8.2 interface. On the left, a tree view shows the project structure with 'GATE' at the top, followed by 'Applications', 'Language Resources', and 'Processing Resources'. Under 'Processing Resources', 'LF_TrainClassification 00029' is selected. The main window is divided into several panes. The 'Loaded Processing resources' pane shows a list of resources including 'ANNE NE Transducer', 'ANNE OrthoMatcher', and 'LF_ApplyClassification 0002A'. The 'Selected Processing resources' pane shows a list of resources including 'Document Reset PR', 'ANNE English Tokeniser', 'ANNE Gazetteer', 'ANNE Sentence Splitter', 'ANNE POS Tagger', 'Annotation Set Transfer 00012', and 'LF_TrainClassification 00029'. Below these panes, there is a dialog box for running the selected resource, 'LF_TrainClassification 00029'. The dialog has a 'Corpus' dropdown set to '<none>' and a 'Runtime Parameters for the "LF_TrainClassification 00029" LF_TrainClassification:' section. This section contains a table of parameters:

Name	Type	Required	Value
algorithmParameters	String		
dataDirectory	URL	✓	file:/home/genevieve/svn/sale/talks/gate-course-jun16/module-3-ml
featureSpecURL	URL	✓	file:/home/genevieve/svn/sale/talks/gate-course-jun16/module-3-ml
inputASName	String		
instanceType	String	✓	trSent
scaleFeatures	ScalingMethod	✓	NONE
sequenceSpan	String		
targetFeature	String		lang
trainingAlgorithm	AlgorithmClassification		LIBSVM_CL

At the bottom of the dialog, there is a 'Run this Application' button. The status bar at the bottom of the window shows 'LF_ApplyClassification 0002A loaded in 0.034 seconds'.

- 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



GATE Developer 8.2-SNAPSHOT build 5490

File Options Tools Help

Messages ANNIE test.xml_00013

Loaded Processing resources

Name	Type
ANNIE NE Transducer	ANNIE NE Transducer
ANNIE OrthoMatcher	ANNIE OrthoMatcher
Annotation Set Transfer 00012	Annotation Set Trans
LF_TrainClassification 00029	LF_TrainClassificati

Selected Processing resources

Name	Type
Document Reset PR	Document Reset PR
ANNIE English Tokeniser	ANNIE English Token
ANNIE Gazetteer	ANNIE Gazetteer
ANNIE Sentence Splitter	ANNIE Sentence Spli
ANNIE POS Tagger	ANNIE POS Tagger
LF_ApplyClassification 0002A	LF_ApplyClassificati

Run "LF_ApplyClassification 0002A"?

Yes No If value of feature is

Corpus: test

Runtime Parameters for the "LF_ApplyClassification 0002A" LF_ApplyClassification:

Name	Type	Required	Value
algorithmParameters	String		
confidenceThreshold	Double	✓	0.0
dataDirectory	URL	✓	file:/home/genevieve/svn/sale/talks/gate-course-jun16/module-3-ml-barbour/classifi
inputASName	String		
instanceType	String	✓	Sentence
outputASName	String		LearningFramework
sequenceSpan	String		
targetFeature	String		lang

Run this Application

Serial Application Editor Initialisation Parameters About...

- 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



GATE Developer 8.2-SNAPSHOT build 5428

File Options Tools Help

Messages ANNIE test

Document statistics Confusion Matrices

Document	Agreed	Total	Observed agreement	Cohen's Kappa
test.xml_00019	232	328	0.71	0.18
Macro summary			0.7100	0.1800
Micro summary	232	328	0.7073	0.1820

Resource Features

Views built!

Corpus editor Initialisation Parameters Corpus Quality Assurance

[Default set]

Key (A)

LearningFramework (B)

Original markups

present in every docum

Annotation Types

Sentence

present in every select

Annotation Features

lang

LF_confidence

LF_target

present in every select

Measures

F-Score Classification

Observed agreement

Cohen's Kappa

- 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

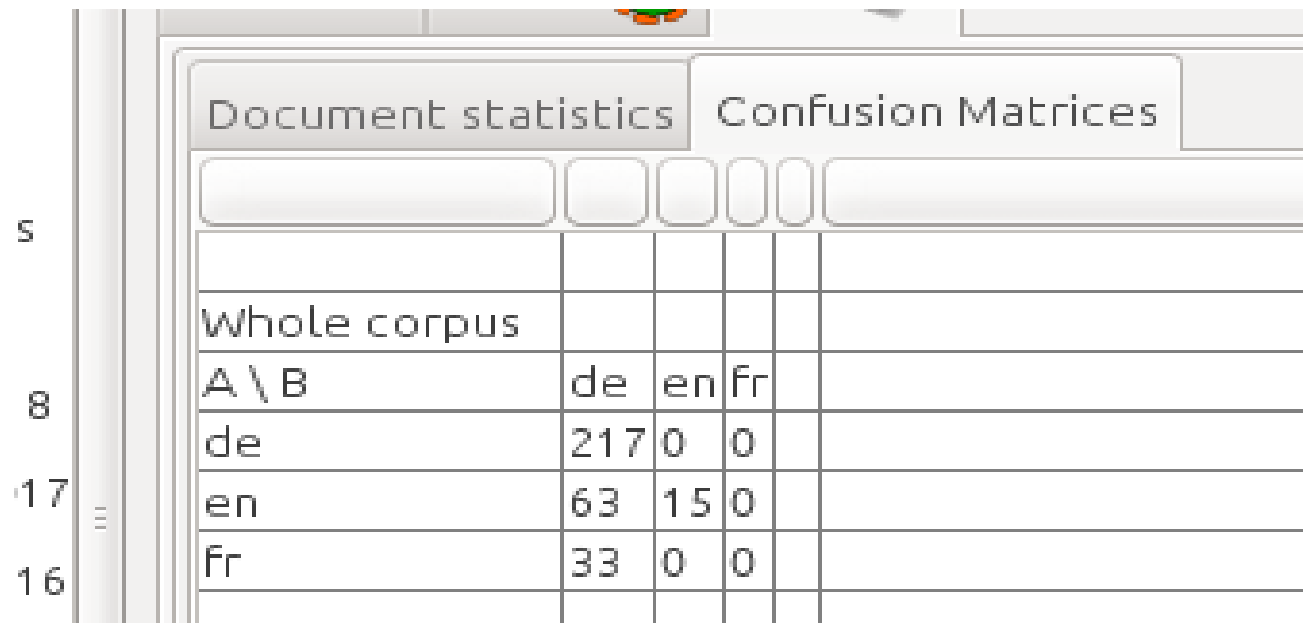
- What do you think about this result? Not bad?
- What do you think of this kappa statistic? (A kappa of over 0.5 is considered good, and over 0.8 excellent.)

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?

A screenshot of the GATE software interface. The "Confusion Matrices" tab is selected. The interface shows a table with the following data:

		Confusion Matrices		
		de	en	fr
Whole corpus				
A \ B		de	en	fr
de		217	0	0
en		63	15	0
fr		33	0	0

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



GATE Developer 8.2-SNAPSHOT build 5428

File Options Tools Help

Messages ANNIE test.xml_00015

Loaded Processing resources

Name	Type
me	
ANNIE NE Transducer	ANNIE NE Transducer
ANNIE OrthoMatcher	ANNIE OrthoMatcher
Annotation Set Transfer 00013	Annotation Set Transfer
LF_ApplyClassification 00020	LF_ApplyClassification

Selected Processing resources

Name	Type
ANNIE Sentence Splitter	ANNIE Sentence Splitter
ANNIE POS Tagger	ANNIE POS Tagger
Annotation Set Transfer 00021	Annotation Set Transfer
LF_EvaluateClassification 00020	LF_EvaluateClassification

Run "LF_EvaluateClassification 0000A"?

Yes No IF value of feature is

Corpus: GATE Corpus_00014

Runtime Parameters for the "LF_EvaluateClassification 0000A" LF_EvaluateClassification:

Name	Type	Required	Value
FeatureSpecURL	URL	✓	file:/home/genevieve/svn/sale/talks/gate-course-jun16/module-3-ml/ml-cl-model/feats.xml
inputASName	String		
instanceType	String	✓	trSent
numberOfFolds	Integer		10
numberOfRepeats	Integer		1
scaleFeatures	ScalingMethod	✓	NONE
sequenceSpan	String		
targetFeature	String		lang
trainingAlgorithm	AlgorithmClassification		WEKA_CL_RANDOM_TREE
trainingFraction	Double		0.6667

Run this Application

Serial Application Editor Initialisation Parameters About...

Resource Features

ANNIE run in 1.176 seconds

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

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?

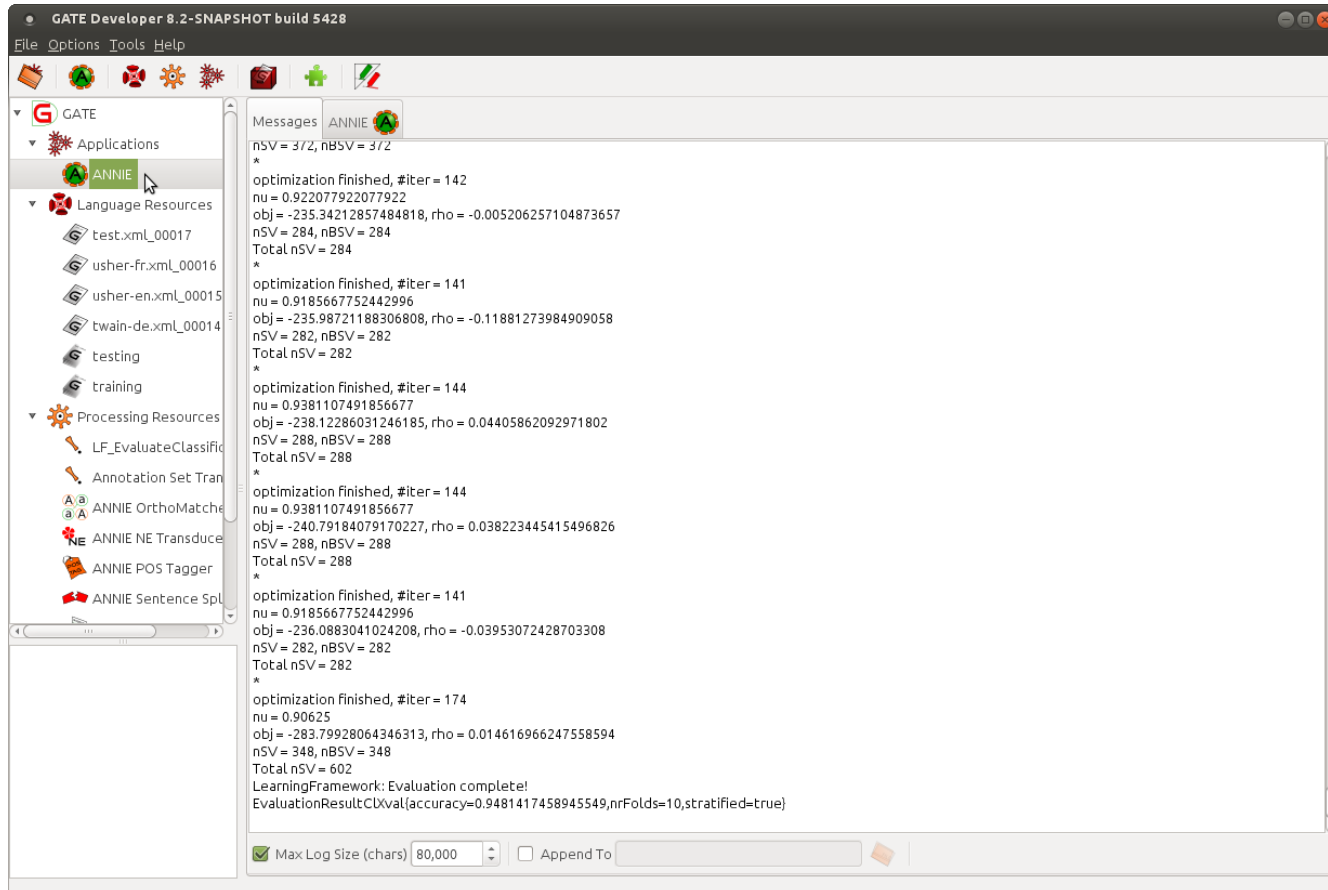


More operations—Evaluation

- 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!



- 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