Part 4:
Practical Sentiment Analysis in GATE

Practical Sentiment Analysis

- In this session, we're going to look at some real-life examples of sentiment analysis in GATE
- You can do sentiment analysis via rule-based or machine-learning-based applications, or both combined
- You can try playing with the applications and tweaking the parameters to see what happens
- Bear in mind that these are only very simple applications, so will not get perfect results!

Rule-based Opinion Mining on Tweets

Why Rule-based?

- Although ML applications are typically used for Opinion Mining, there are advantages to using a rule-based approach when training data isn't easily available
- For example, working with multiple languages and/or domains, or when you don’t have lots of training data
- Rule-based system is more easily adaptable
- Novel use of language and grammar in social media makes ML hard
- ML struggles to deal with the nitty-gritty linguistic detail

GATE Components for Opinion Mining

- ANNIE or TwitIE for language pre-processing and NER
- (Optional) term recognition using TermRaider
- Sentiment gazetteer lookup
- JAPE language analysis grammars
- JAPE opinion detection grammars
- (Optional) aggregation of opinions

Basic approach for opinion finding

- Find sentiment-containing words in a linguistic relation with terms/entities (opinion-target matching)
  - e.g. “life flourishing in Antarctica”
- Dictionaries give a starting score for sentiment words
- Use a number of linguistic sub-components to deal with issues such as negatives, adverbial modification, swear words, conditionals, sarcasm etc.
- Modify the score appropriately
- Find the opinion holders and correctly match them to the opinions

A positive sentiment list

awesome	category=adjective	score=0.5
beaming	category=adjective	score=0.5
belonging	category=noun	score=0.5
benefic	category=adjective	score=0.5
benevolently	category=adverb	score=0.5
caring	category=noun	score=0.5
charitable	category=adjective	score=0.5
charm	category=verb	score=0.5

A negative sentiment list

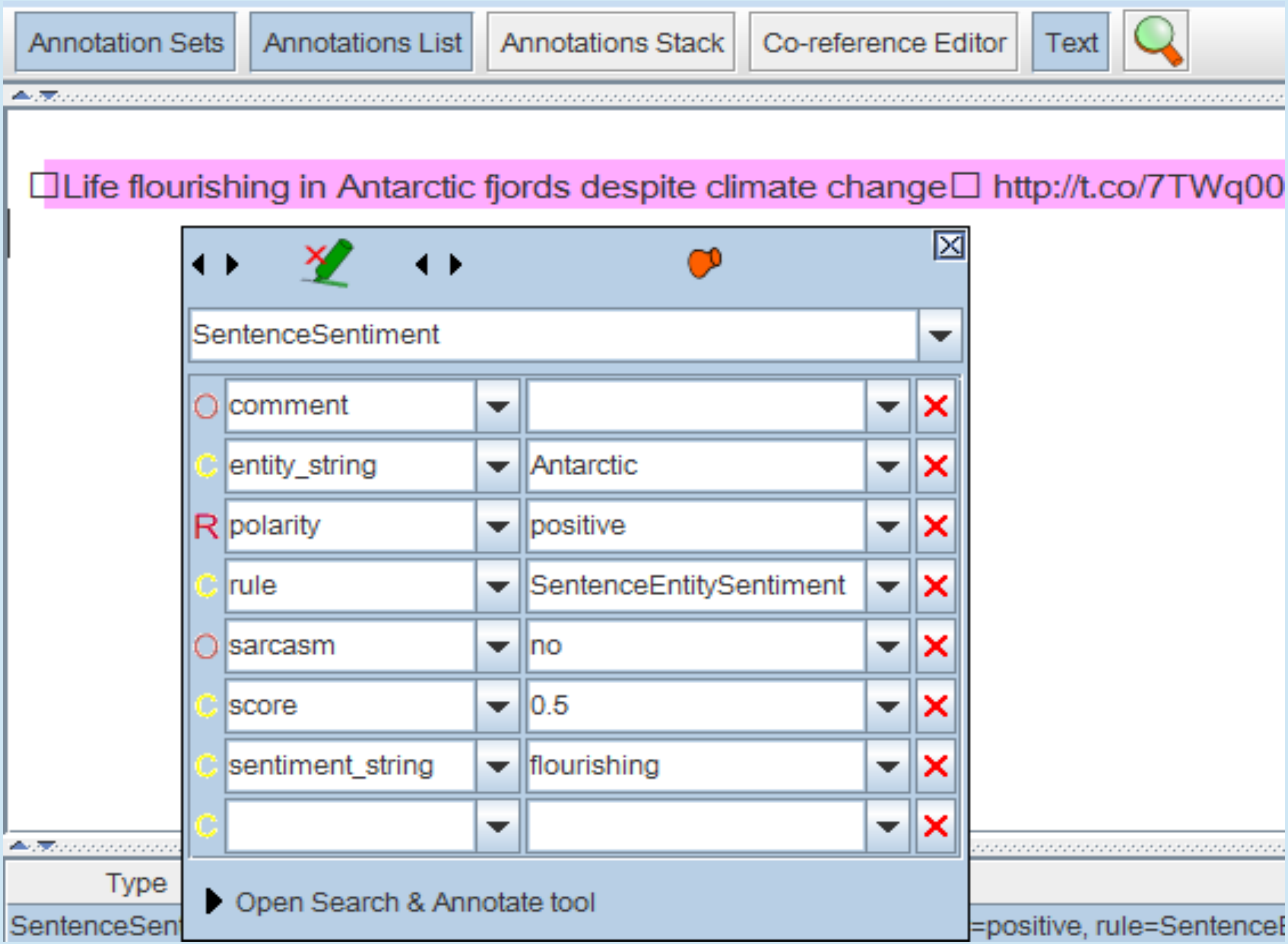
Examples of phrases following the word "go":

- down the pan
- down the drain
- to the dogs
- downhill
- pear-shaped

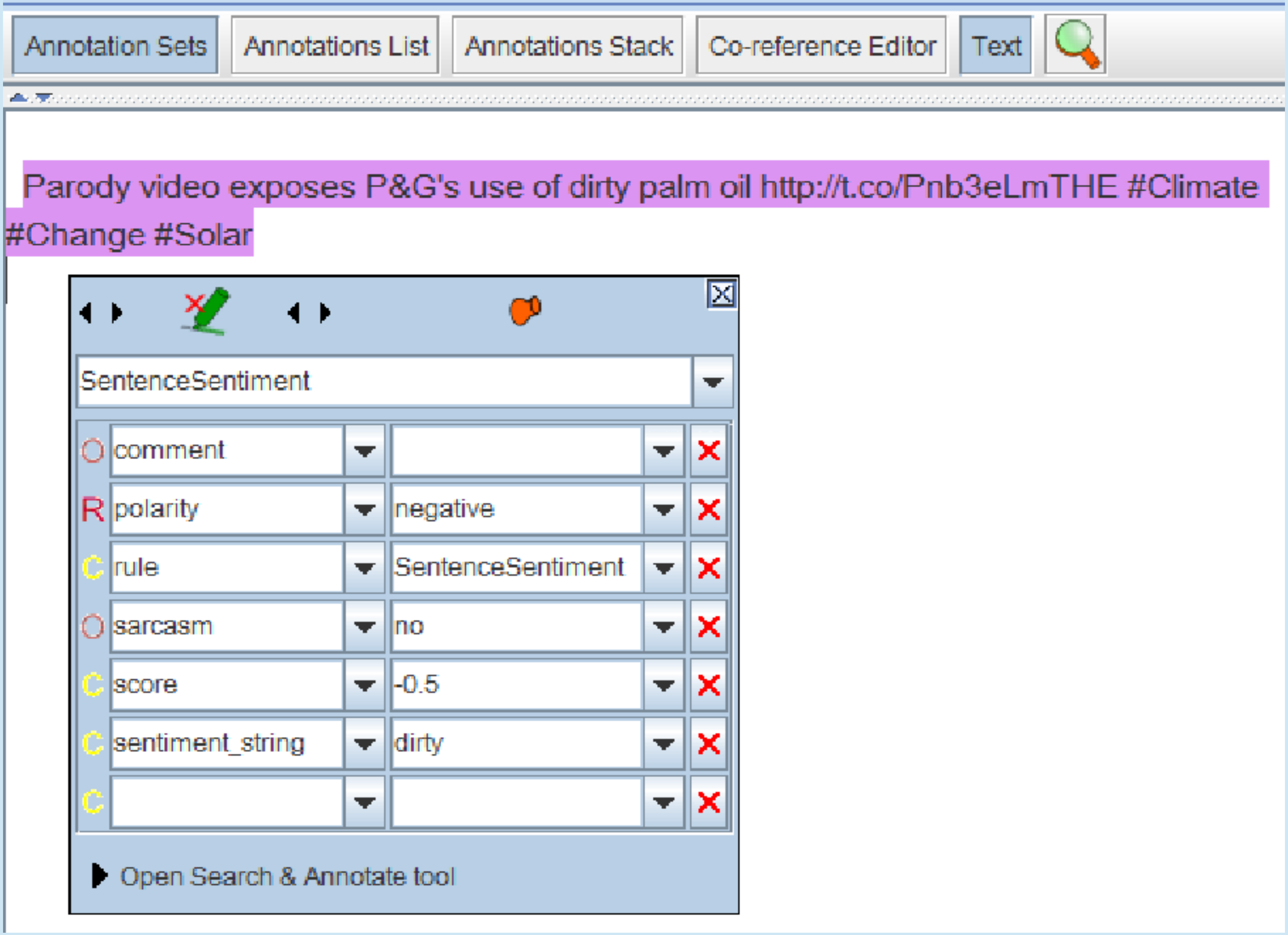
Opinion scoring

- Sentiment gazetteers (developed from sentiment words in WordNet) have a starting “strength” score
- These get modified by context words, e.g. adverbs, swear words, negatives and so on.
  - *The film was awesome --> The film was **really** amazing*.
  - *The film was awful --> The film was **absolutely** awful.*
  - *The film was good --> The film was **not so** good.*
- Swear words modifying adjectives count as intensifiers
  - *The film was good --> The film was **damned** good.*
- Swear words on their own are classified as negative
  - ***Damned** politicians.*

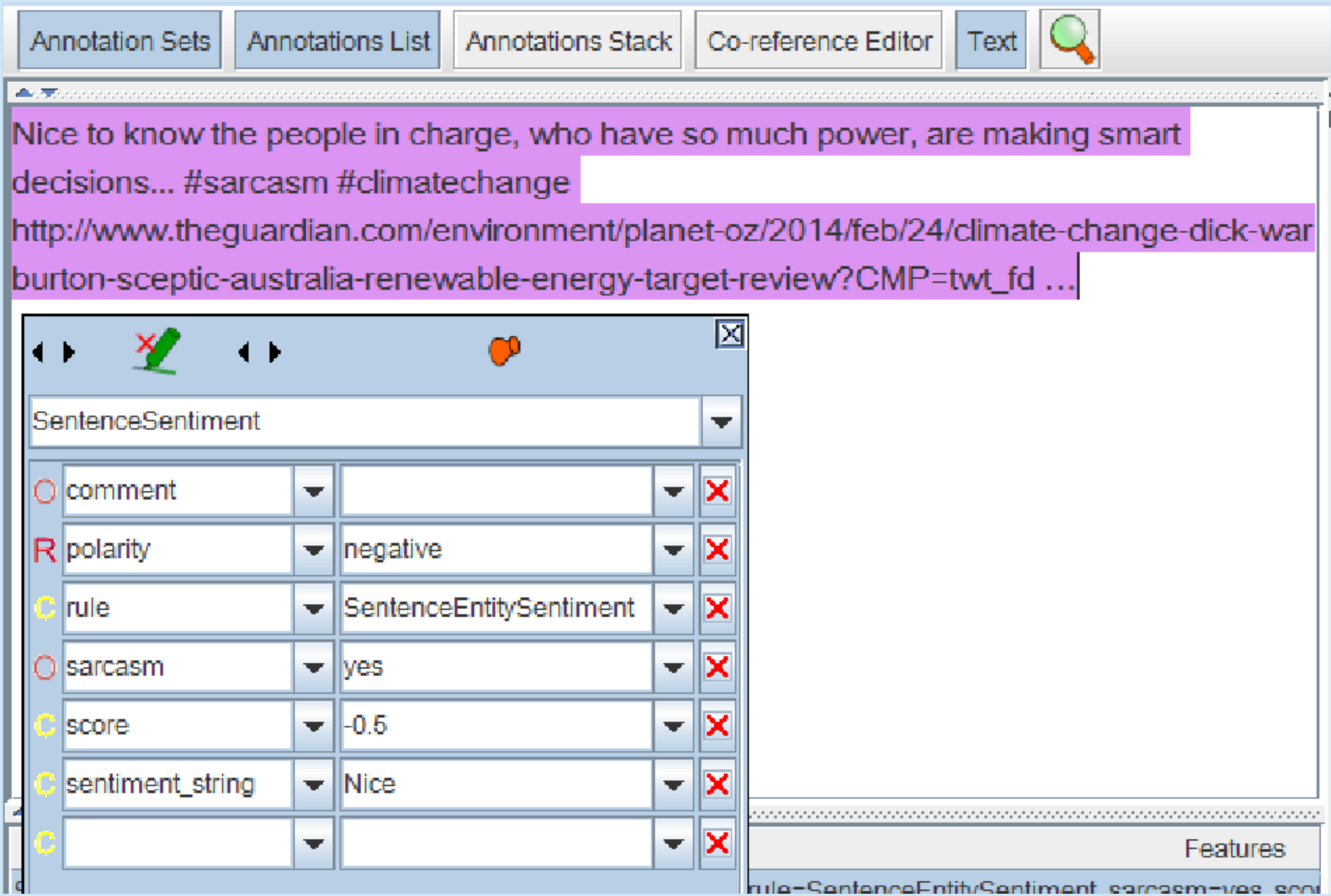
A positive tweet in GATE



A negative tweet in GATE



A Sarcastic Tweet in GATE



Creating a corpus

- First step is to create a corpus of tweets
- We can use the Twitter Streaming API to collect all the tweets over a particular period according to various criteria (e.g. use of certain hashtags, mention of various political parties etc.)
- Collect the tweets in JSON and import directly into GATE
- This gives us lots of additional twitter metadata, such as the date and time of the tweet, the number of followers of the person tweeting, the location and other information about the person tweeting, and so on
- This information may be useful for querying the data later (see the Applications section later)

Hands-on: Analysing tweets

- Load the TwitIE application if you don’t have it already
- Unzip generic-opinion-mining.zip and load the file application.xgapp as an application
- Add TWITIE to the very beginning of this application
- Load the test file testTweets_small.txt from the hands-on corpora directory and add to a corpus
- Run the application on the document and check the results
  - HINT: make sure you run the app called english-om and not TwitIE

Hands-on analysing tweets (2)

- Look at the SentenceSentiment annotation and check its features
- Try modifying the sentiment-words gazetteer to improve the results (click on the gazetteer to view it and then edit the lists in it to add new words
- Save and reinitialise it, and then also right click on sentiment-words-extended-gaz and select “Remove cache and reinitialise”
- A simple improvement might be to try to create a Sentiment annotation for the tweet “i heart u!”
    - HINT: add the word “heart” to the list called “positive.lst”

Sarcasm is a part of British culture

- So much so that the BBC has its own webpage on sarcasm, designed to teach non-native English speakers how to be sarcastic successfully in conversation

Some common examples of sarcasm

Remember to judge when and with whom to be sarcastic - you can offend people with inappropriate use of this language.

After something bad or annoying happens:
Oh terrific / great / brilliant! That's just what I need.

After something unsurprising happens:
Well what a surprise!

After somebody makes a mistake:
Oh nice one!
Oh well done!

After someone says something obvious:
No?! Really? You're quick / clever!

BBC sarcasm quiz

Sorry, I think I've broken your screwdriver.

Was that thunder I just heard?

You know, I think Marie and Jack might be seeing each other.

And this car is now under $50,000!

You shouldn't use prepositions at the end of sentences.

Erm, Dad... I've run out of money.

Congratulations! Susie told me you were pregnant.

Wow, what a fantastic deal.

Well if you heard it from Susie it must be true.

Oh yes, I forgot you were the English grammar expert.

Gosh you're quick. They've been dating for months.

Oh what a surprise. How much do you need?

Oh brilliant. So much for tennis this afternoon!

Oh well done! Let me have a look at it.

check

How do you know when someone is being sarcastic?

- Use of hashtags in tweets such as #sarcasm, emoticons etc.
- Large collections of tweets based on hashtags can be used to make a training set for machine learning
- But you still have to know which bit of the tweet is the sarcastic bit

Man , I hate when I get those chain letters & I don't resend them , then I die the next day .. #Sarcasm

To the hospital #fun #sarcasm

What does sarcasm do to polarity?

- In general, when someone is being sarcastic, they're saying the opposite of what they mean
- So as long as you know which bit of the utterance is the sarcastic bit, you can simply reverse the polarity
- To get the polarity scope right, you need to investigate the hashtags: if there's more than one, you need to look at any sentiment contained in them.

Identifying the scope of sarcasm

*I am **not happy** that I woke up at 5:15 this morning. #**great**start #sarcasm*

- negative sentiment + positive hashtag + sarcasm hashtag
- The positive hashtag becomes negative with sarcasm

*You are **really mature**. #**lying** #sarcasm*

- positive sentiment + sarcasm hashtag + sarcasm hashtag
- The positive sentiment is turned negative by both sarcasm hashtags

- Not all sarcastic tweets are negative.
- Can you think of a positive example?

What if you don't have a hashtag or other indicator?

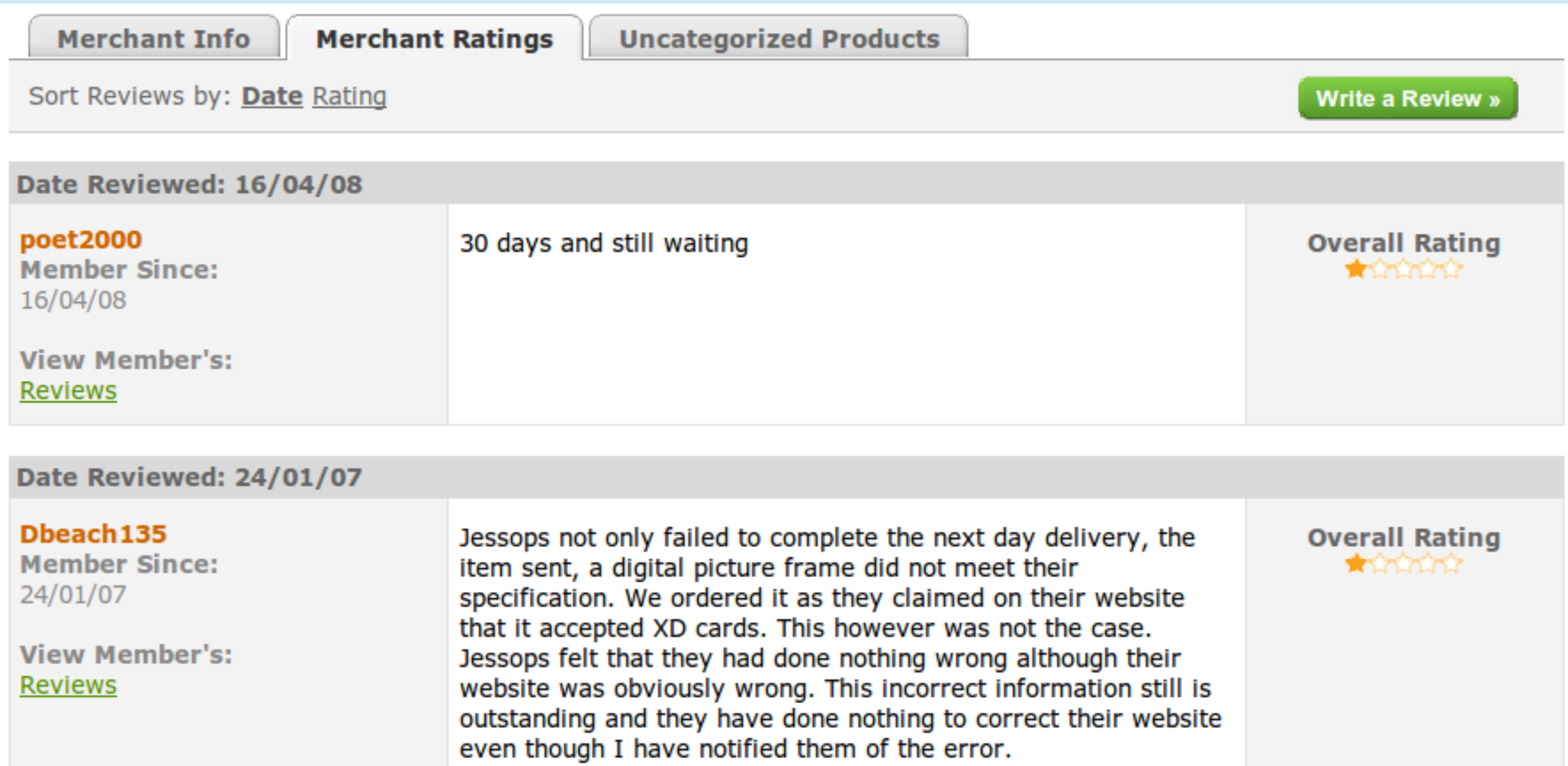
- Look for word combinations with opposite polarity, e.g. “rain” or “delay” plus “brilliant”
  - *Going to the dentist on my weekend home. Great. I'm totally pumped. #sarcasm*
- Inclusion of world knowledge / ontologies can help (e.g. knowing that people typically don't like going to the dentist, or that people typically like weekends better than weekdays.
- It's an incredibly hard problem and an area where we expect not to get it right that often

Machine Learning-based applications

Machine Learning for Sentiment Analysis

- ML is an effective way to classify opinionated texts
- We want to train a classifier to categorize free text according to the training data.
- Good examples are consumers' reviews of films, products, and suppliers.
- Sites like www.pricegrabber.co.uk show reviews and an overall rating for companies: these make good training and testing data
- We train the ML system on a set of reviews so it can learn good and bad reviews, and then test it on a new set of reviews to see how well it distinguishes between them
- We give an example of a real application and some related hands-on for you to try

Examples of consumer reviews



Merchant Info | Merchant Ratings | Uncategorized Products

Sort Reviews by: Date Rating

Write a Review »

Date Reviewed: 16/04/08

poet2000 **Member Since:** 16/04/08 **View Member's:** Reviews	30 days and still waiting	**Overall Rating**

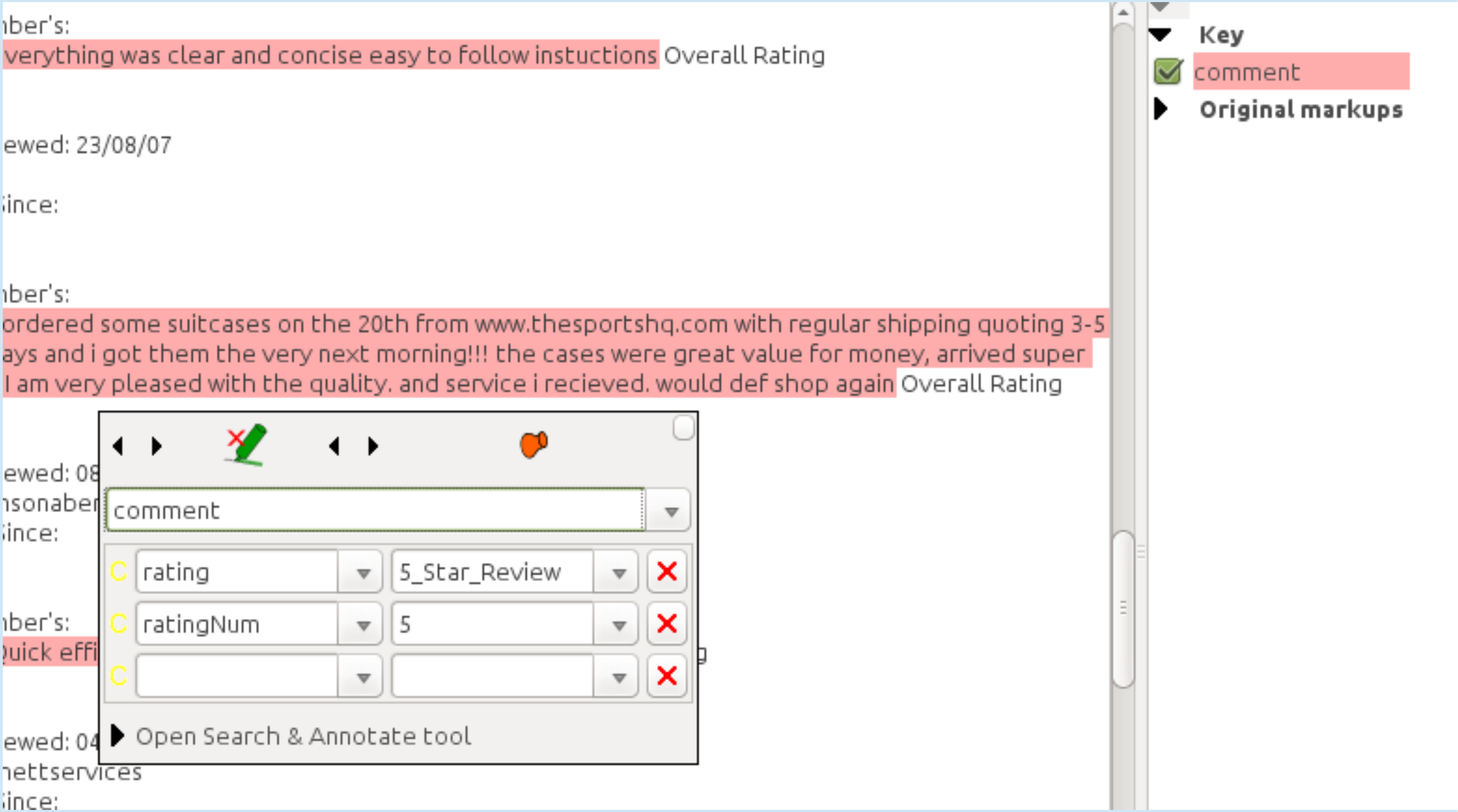
Date Reviewed: 24/01/07

Dbeach135 **Member Since:** 24/01/07 **View Member's:** Reviews	Jessops not only failed to complete the next day delivery, the item sent, a digital picture frame did not meet their specification. We ordered it as they claimed on their website that it accepted XD cards. This however was not the case. Jessops felt that they had done nothing wrong although their website was obviously wrong. This incorrect information still is outstanding and they have done nothing to correct their website even though I have notified them of the error.	**Overall Rating**

Preparing the corpus

- Corpus of 40 documents containing 552 company reviews.
- Each review has a 1- to 5-star rating.
- We pre-processed these in GATE to label each review with a comment annotation with a rating feature (free manual annotation!)
- In ML terms:
  - instance = *comment* annotation
  - class = *rating* feature on the *comment* annotation
  - attributes = NLP features of the underlying text
- We will keep the spans of the comment annotations and use ML to classify them with the *rating* feature

Annotated review



Developing the training application

- We will develop an application that runs a set of NLP components to provide ML instance attributes, and trains the classifier
- Load the ANNIE, Tools, and Learning Framework plugins
- Create a new corpus called “training” and populate it from the directory **ml-exercise/corpora/training** in the hands-on material
- Let’s look at the config file **ml-exercise/feats.xml**

Feature Specification

```
<ML-CONFIG>
<NGRAM>
<NUMBER>1</NUMBER>
<TYPE>Token</TYPE>
<FEATURE>string</FEATURE>
</NGRAM>
</ML-CONFIG>
```

- The feature specification indicates we should use every Token string in the instance
- The “number” indicates to use unigrams

Building the training application (1)

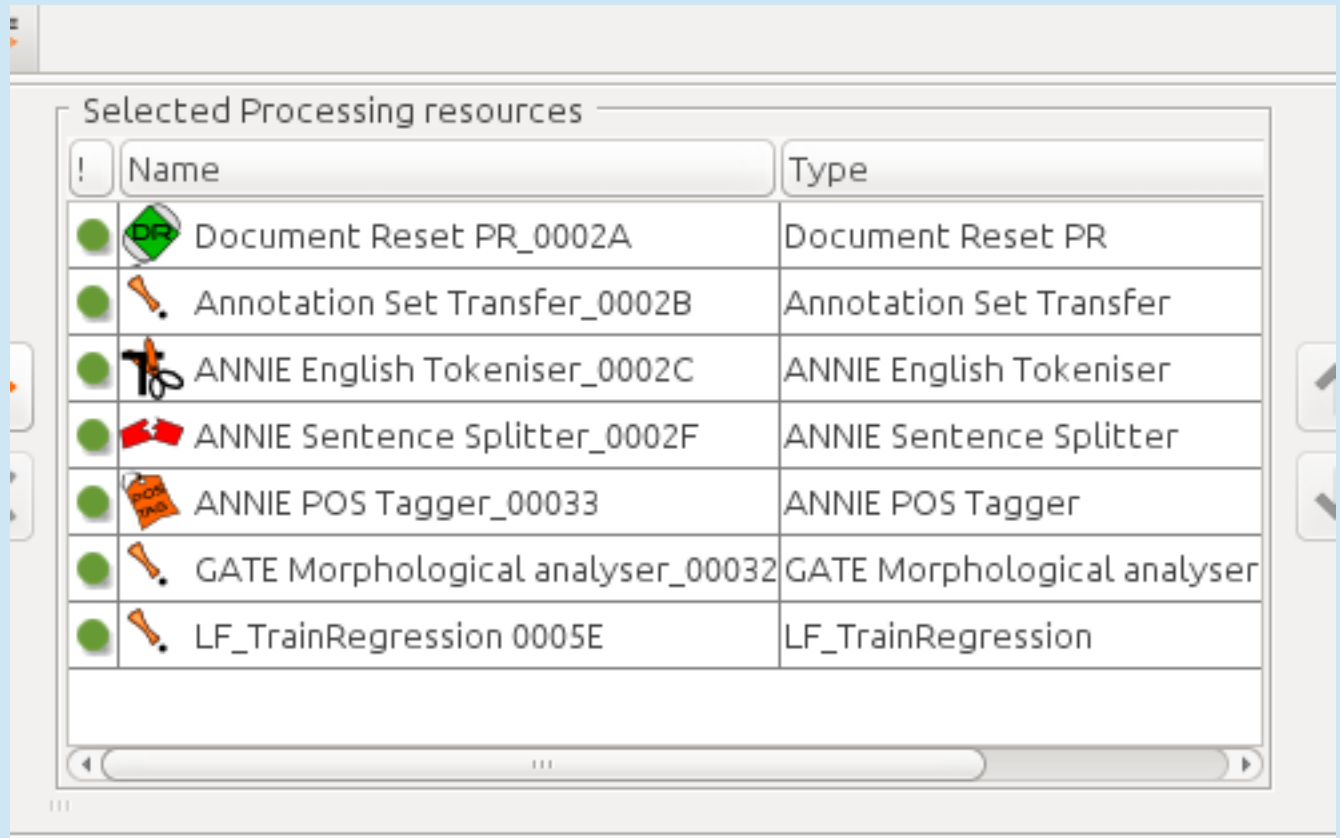
- Create the following PRs with the default init parameters:
  - Document Reset PR
  - Annotation Set Transfer
  - ANNIE English Tokeniser
  - ANNIE Sentence Splitter
  - ANNIE POS Tagger
  - GATE Morphological Analyser
- LF_TrainRegression
  - LF_ApplyRegression
  - Create a new Conditional Corpus Pipeline.

Building the application (2)

- We want to copy the comment annotations to the default annotation set to provide the ML instances and classes, but we don't want to remove the Key annotations
- Add the PRs to the pipeline & set some runtime parameters
  - Document Reset (default parameters)
  - Annotation Set Transfer:
    - annotationTypes = empty list (copy all)
    - copyAnnotations = true
    - inputASName = “Key”
    - outputASName & textTagName must be blank

Building the application (3)

- Add the remaining loaded PRs to the pipeline
  - English tokeniser
  - Sentence splitter
  - POS tagger
  - Morphological analyser
  - LF_TrainRegression



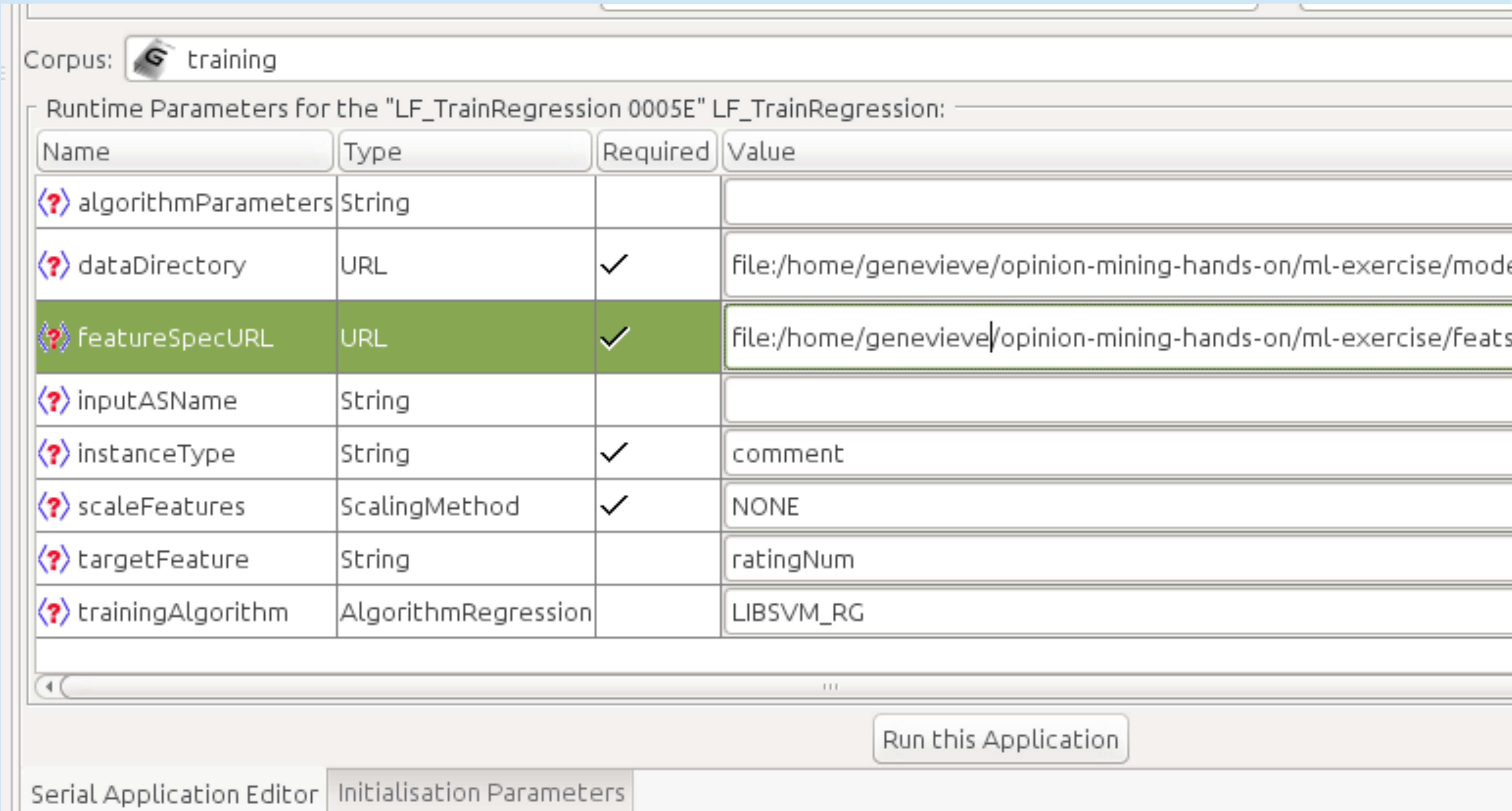
Learning Framework Parameters

- algorithmParameters: set to “-c 100” (explained in the ML module)
- dataDirectory is where the model will be saved. Create an empty directory and specify it here
- featureSpecURL is the feature specification file we inspected earlier
- inputASName is the default annotation set (blank)
- instanceType is the name of the instance annotation type (“comment”)
- scaleFeatures can be ignored
- targetFeature is “ratingNum” (the numeric version)
- trainingAlgorithm is LIBSVM_RG

Algorithm and Target

- We are using a regression algorithm to do this task, because we are learning to predict numbers
- You could do this as a classification task by treating the ratings as words (using the “rating” feature), but numbers contain more information that words. We know that three is bigger than one and smaller than five
- By using regression we can take into account that where the target is five, four is less wrong than one
- LIBSVM_RG uses a support vector machine to perform regression

Learning Framework Parameters



Running the Training Application

- Run it on the training corpus (this should take less than 1 minute)
- The classifier's model is stored in the directory you indicated. The model is stored in text files, but they are not meant to be human-readable.

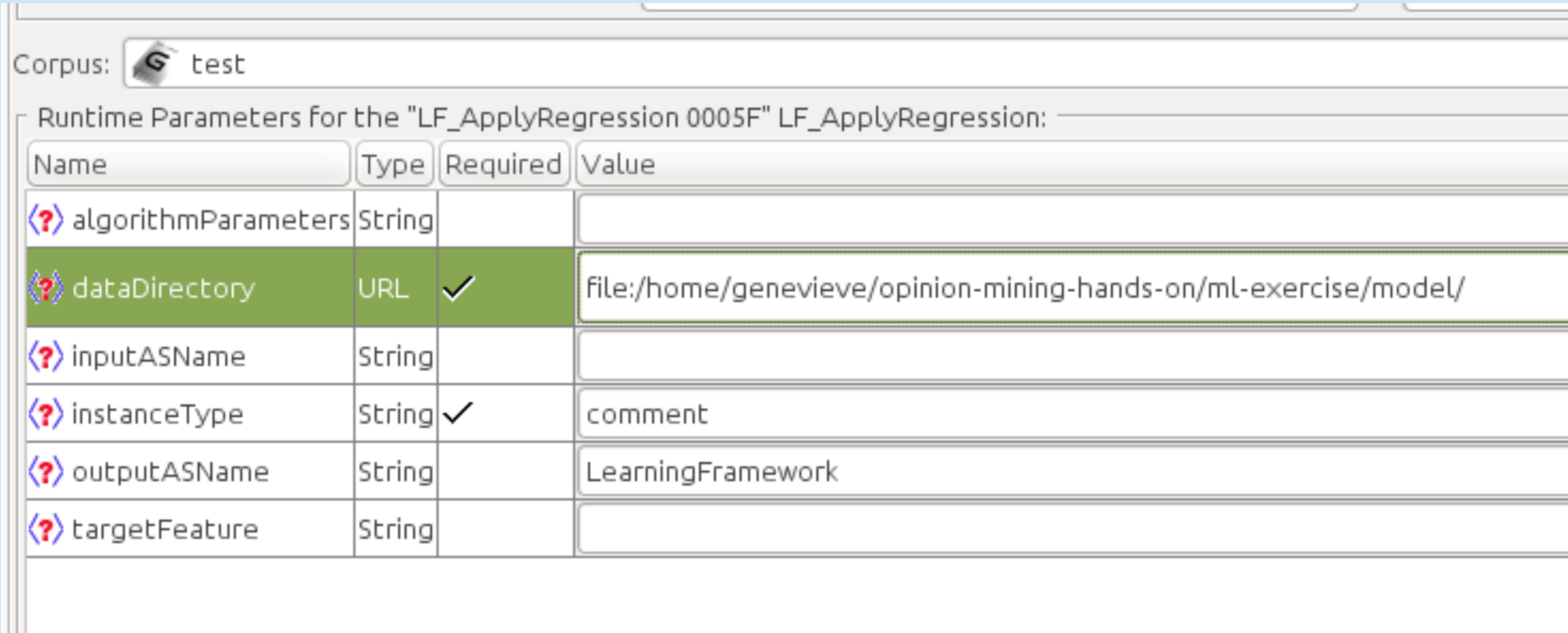
Applying the training model (1)

- Create a “testing” corpus and populate it from the **corpora/testing** directory.
- To apply the model, we need to have comment annotations *without* rating features on the default AS. These will give us the instances to classify. A simple JAPE Transducer can do this.
- Load the grammar resources/grammar/copy_comment_spans.jape.
- Insert the grammar in the pipeline after the AS Transfer PR.
- Set the transducer parameters:
  - inputASName = “Key”
  - outputASName = “”

Applying the training model (2)

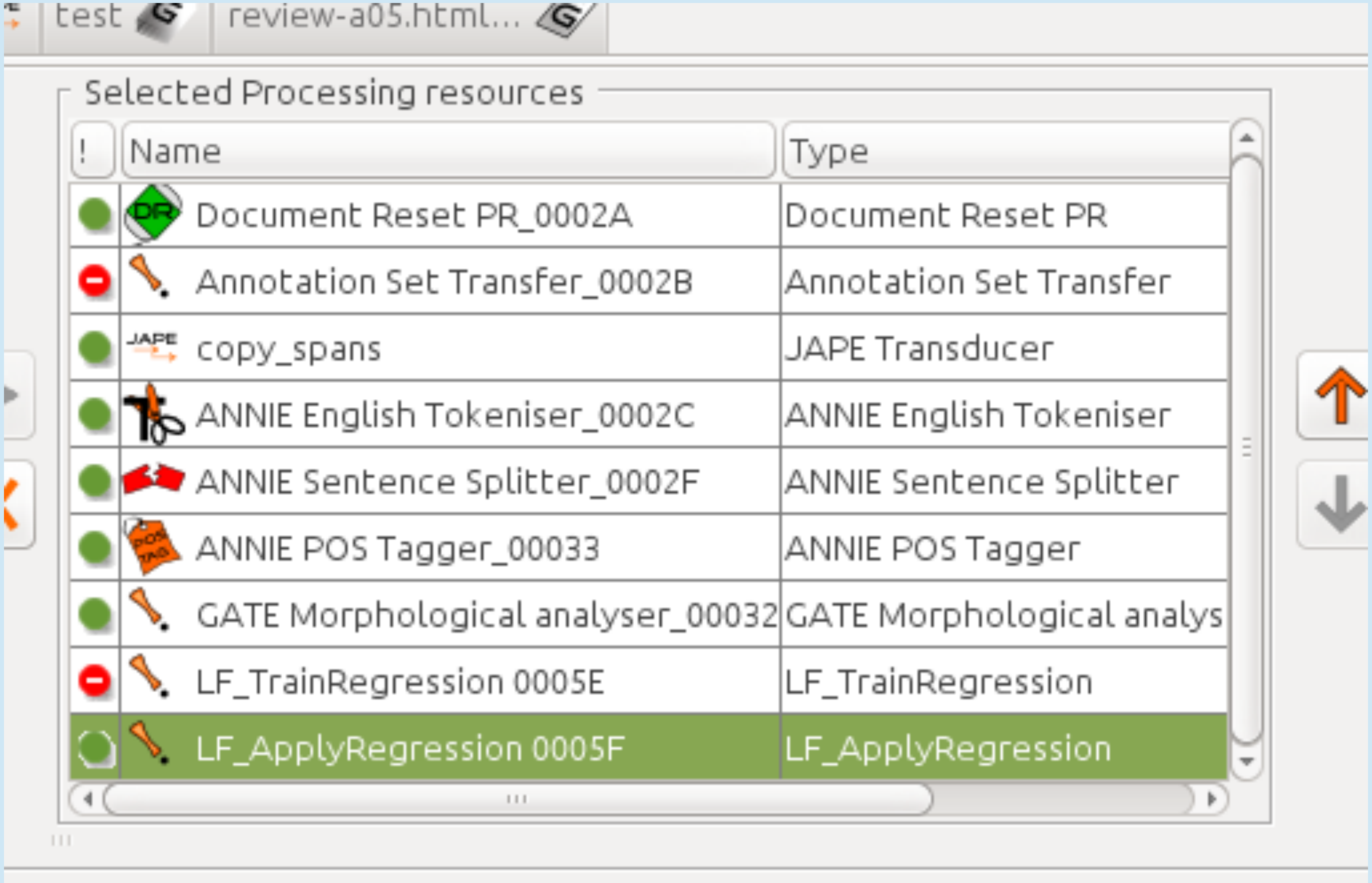
- Set the AS Transfer PR's run-mode to “no” (red light)
- Set the LF_TrainRegression PR's run-mode to “no”
- Add the LF_ApplyRegression PR
- The classifier will get instances (*comment* annotations) and attributes (other annotations' features) from the default AS and put instances with classes (*rating* features) in the Output AS.

LF_ApplyRegression Parameters



- dataDirectory is where you saved your model during training`
- instanceType is “comment”, as previously
- outputASName defaults to LearningFramework, which helps us to clearly see what has been created
- targetFeature if left blank defaults to the same one as in training

Applying the training model (3)

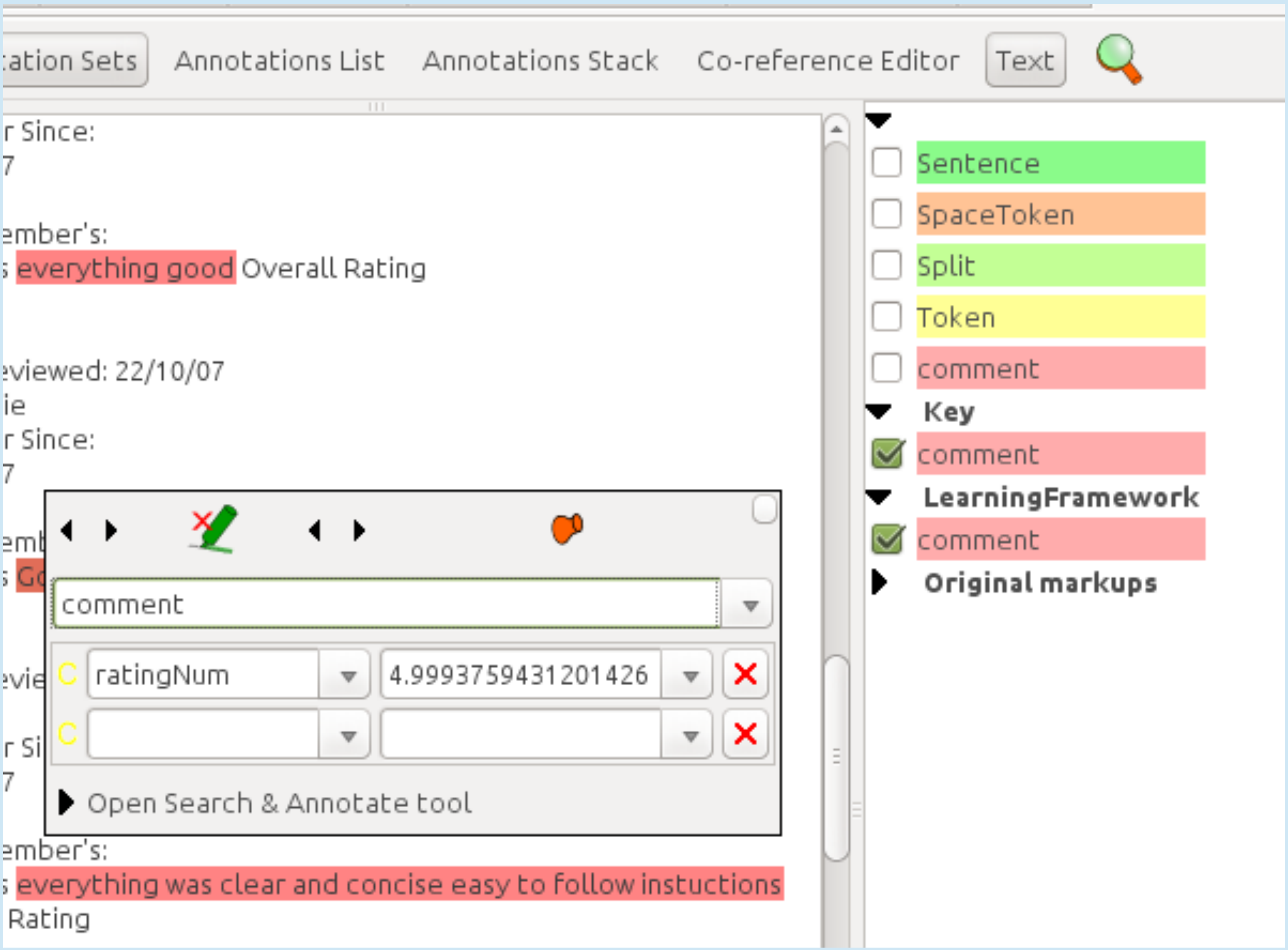


- Run the pipeline on the testing corpus

Applying the training model (3)

- Open a few documents and inspect the “comment” annotations:
  - “Key” AS = user ratings (instances and correct classes)
  - default AS = instances & attributes but no classes
  - “LearningFramework” AS = instances with ratingNum values generated by ML

Annotation Results



Applying the training model (4)

- Note that the values are real numbers, not integers, so Corpus QA will not work
- Create a JAPE transducer PR from the numeric-to-string.jape file
- Add it to the end of the application and set both inputASName and outputASName to “LearningFramework”
- Run the application again: the output annotations now have additional “rating” features with values “1_Star_Review”, “2_Star_Review”, etc., so the results can be measured with Corpus QA and other tools

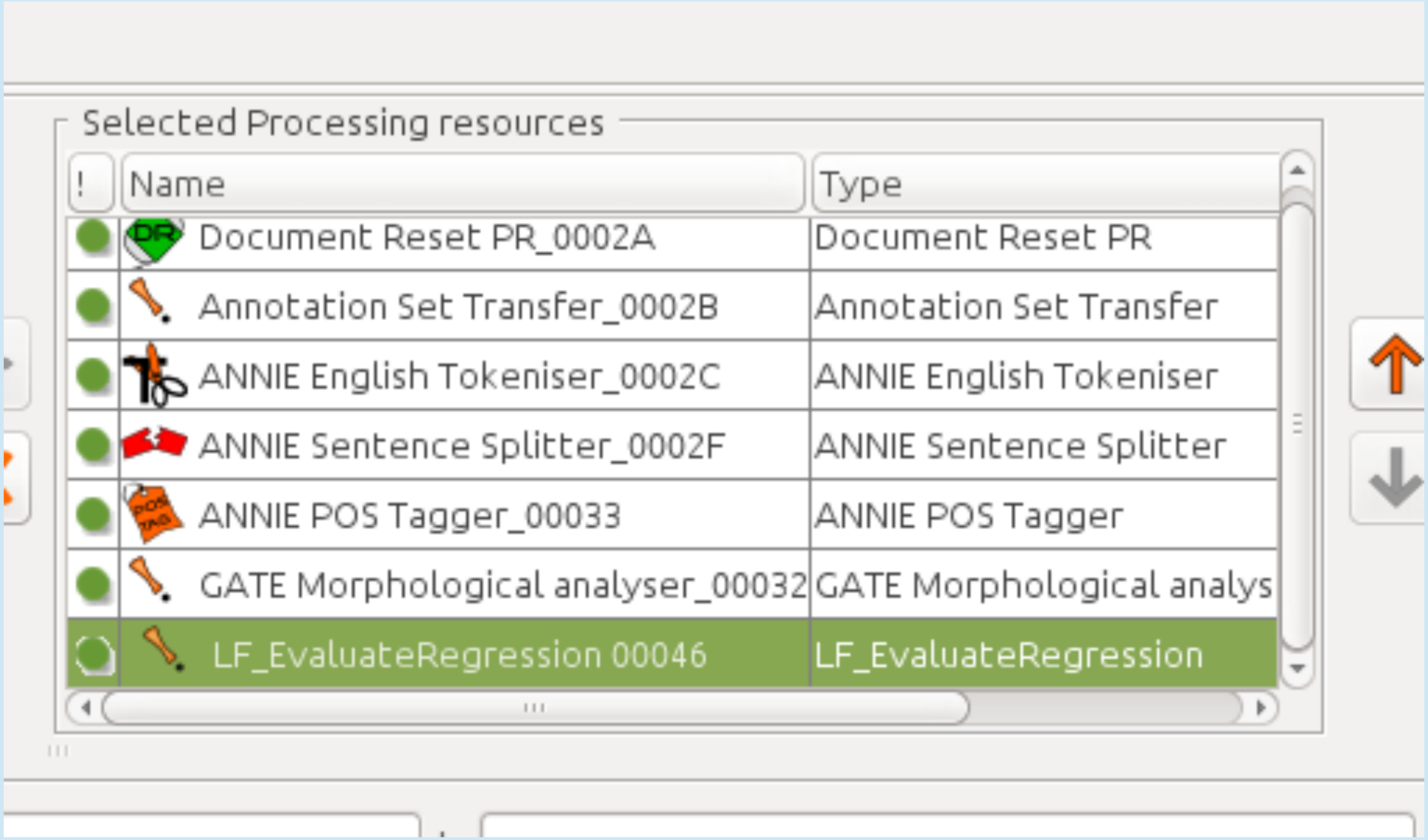
Cross-validation

- Cross-validation is a standard way to “stretch” the validity of a manually annotated corpus, because it enables you to test on a larger number of documents
- The 5-fold averaged result is more significant than the result obtained by training on 80% of the same corpus and testing on 20% once.

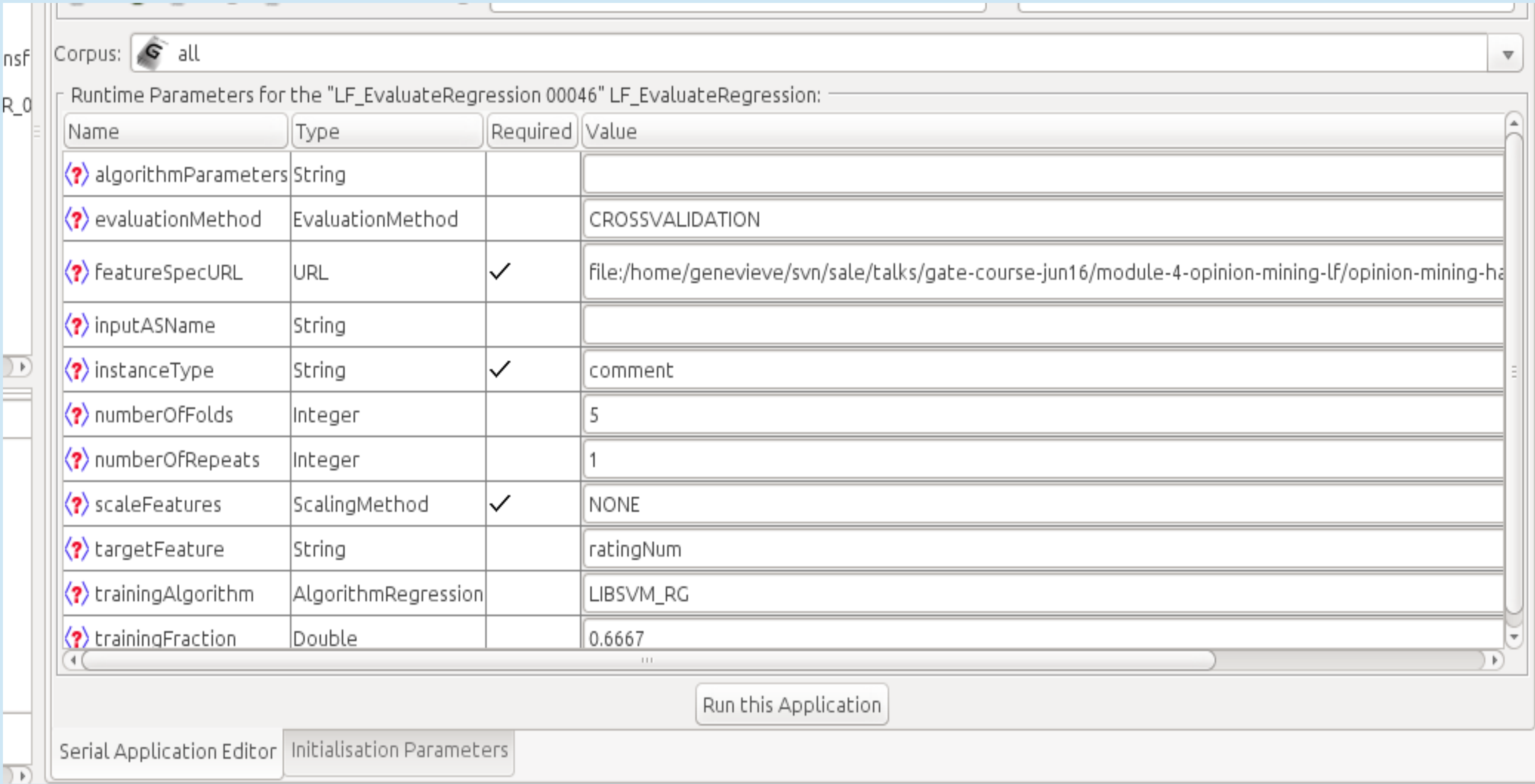
LF_EvaluateRegression

- The LF_EvaluateRegression PR will automatically split the corpus into 5 parts, and then
    - train on parts 1,2,3,4; apply on part 5;
    - train on 1,2,3,5; apply on 4;
    - train on 1,2,4,5; apply on 3;
    - train on 1,3,4,5; apply on 2;
    - train on 2,3,4,5; apply on 1;
  - and average the results. For regression, the PR will print the RMSE (root mean square error).

LF_EvaluateClassification



LF_EvaluateRegression Parameters



Name	Type	Required	Value
algorithmParameters	String		
evaluationMethod	EvaluationMethod		CROSSVALIDATION
featureSpecURL	URL	✓	file:/home/genevieve/svn/sale/talks/gate-course-jun16/module-4-opinion-mining-lf/opinion-mining-ha
inputASName	String		
instanceType	String	✓	comment
numberOfFolds	Integer		5
numberOfRepeats	Integer		1
scaleFeatures	ScalingMethod	✓	NONE
targetFeature	String		ratingNum
trainingAlgorithm	AlgorithmRegression		LIBSVM_RG
trainingFraction	Double		0.6667

Summary

- Simple examples of rule-based and ML methods for creating OM applications
- How to work with tweets in GATE
- Examples of how deeper linguistic information can be useful
- In the final part of this tutorial, we'll look at some real-life applications fo sentiment and social media analysis

Suggestions for further ML experiments...

Suggestions...

- The config file can be copied and edited with any text editor.
- Try n-grams where n>1
  - Change <NUMBER> in the config
  - Usually this is slower, but sometimes it improves quality
- Adjust the cost (-c value)
  - Increasing it may increase correct classifications, but can lead to overfitting.

Suggestions...

- Try using other features
  - *Token.string*, *Token.category*, or combinations of these with *Token.root* and *Token.orth*
- You could even include other ANNIE PRs in the pipeline and use Lookup or other annotation types.
- You need to create the same attributes for training and application.
- If an instance does not contain at least one attribute (annotation+feature specified in the config file), the ML PR will throw a runtime exception, so it's a good idea to keep a Token.string unigram in the configuration.

More information

- GATE website http://gate.ac.uk
- GATE cloud (demos etc) http://cloud.gate.ac.uk
- GATE blog (lots of interesting posts about our social media analysis tools): http://gate4ugc.blogspot.co.uk/
- DecarboNet project: monitoring sentiment about climate change in social media http://www.decarbonet.eu
- COMRADES project: social media analysis in disasters http://www.comrades-project.eu
- Political Futures Tracker: https://gate.ac.uk/projects/pft/
- SoBigData project (many social media applications): http://www.sobigdata.eu

Some publications (more on the GATE website)

- D. Maynard, I. Roberts, M. A. Greenwood, D. Rout and K. Bontcheva. A Framework for Real-time Semantic Social Media Analysis. Web Semantics: Science, Services and Agents on the World Wide Web, 2017
- K. Bontcheva, L. Derczynski, A. Funk, M.A. Greenwood, D. Maynard, N. Aswani. TwitIE: An Open-Source Information Extraction Pipeline for Microblog Text. Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2013).
- D. Maynard, K. Bontcheva, I. Augenstein. Natural Language Processing for the Semantic Web. Morgan and Claypool, December 2016. ISBN: 9781627059091
- D. Maynard and K. Bontcheva. Challenges of Evaluating Sentiment Analysis Tools on Social Media. In Proc. of Language Resources and Evaluation Conference (LREC), May 2016, Portoroz, Slovenia.

Acknowledgements

Current work supported by:

- the European Union/EU under the Information and Communication Technologies (ICT) theme of the 7th Framework and H2020 Programmes for R&D
  - SoBigData (654024) http://www.sobigdata.eu
  - COMRADES (687847) http://www.comrades-project.eu
- Nesta http://nesta.org.uk