ClearTK Tutorial

Steven Bethard

University of Colorado Boulder

Mon 11 Jun 2012

What is ClearTK?

Framework for machine learning in UIMA components

- Feature extraction from CAS
- Common classifier interface across:
  - OpenNLP Maxent, Mallet, GRMM, libSVM, SVMlight
- Training and loading classifiers from JARs

UIMA wrappers for non-UIMA components

- Berkeley parser
- ClearParser
- MaltParser
- Stanford CoreNLP

In-house machine learning components, e.g. for TimeML

What is ClearTK?

Framework for machine learning in UIMA components

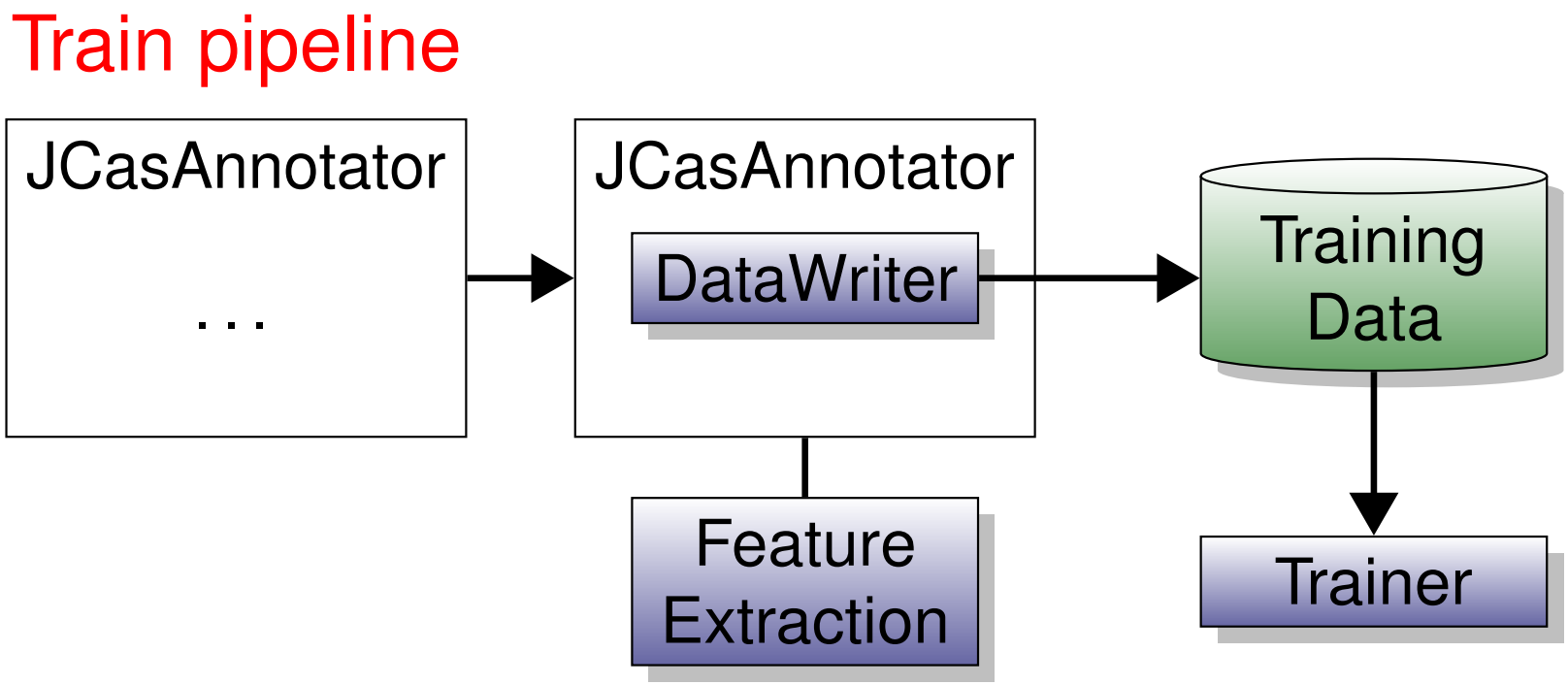
- Feature extraction from CAS
- Common classifier interface across:
  - OpenNLP Maxent, Mallet, GRMM, libSVM, SVMlight
- Training and loading classifiers from JARs

UIMA wrappers for non-UIMA components

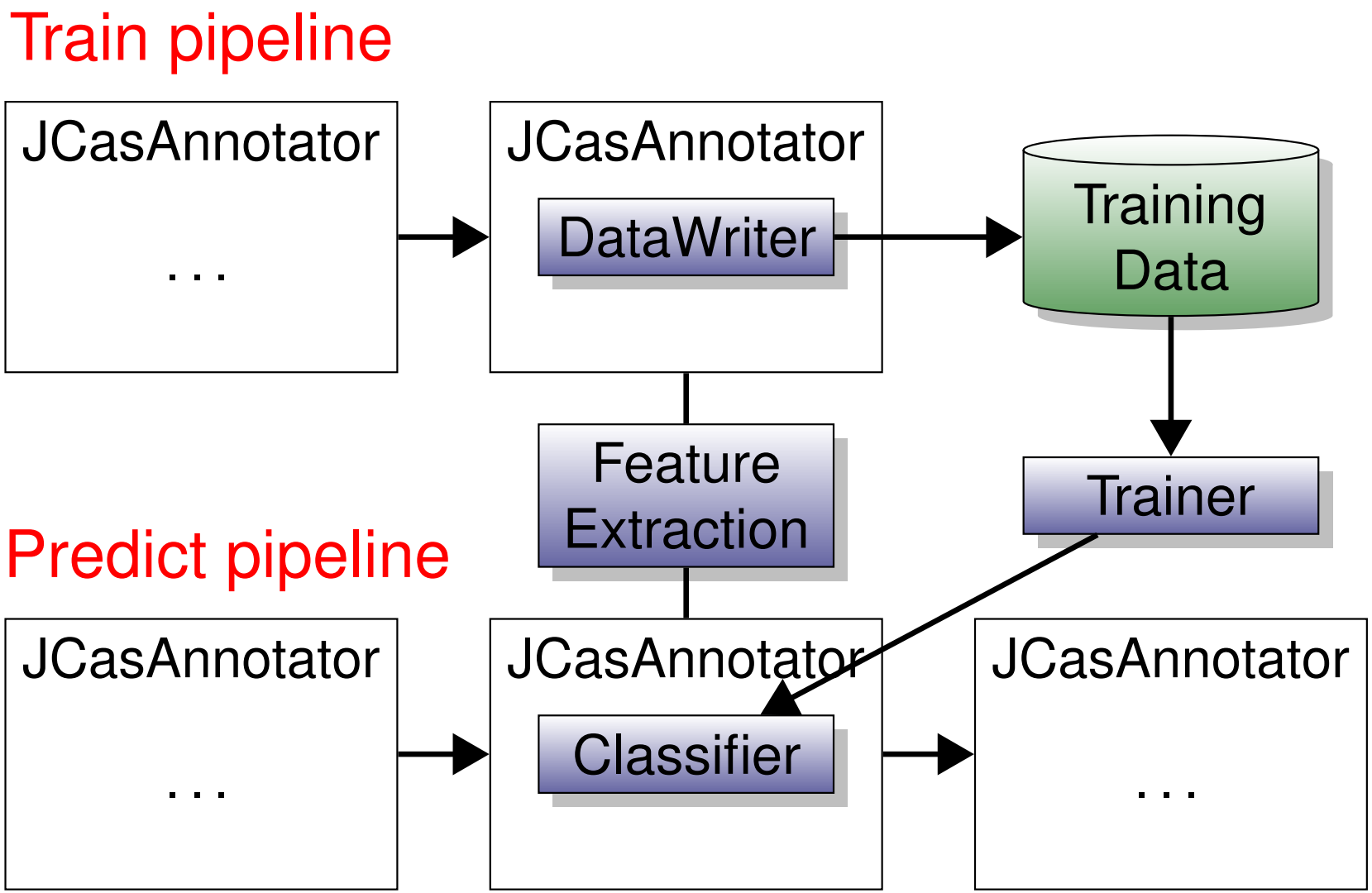
- Berkeley parser
- ClearParser
- MaltParser
- Stanford CoreNLP

In-house machine learning components, e.g. for TimeML

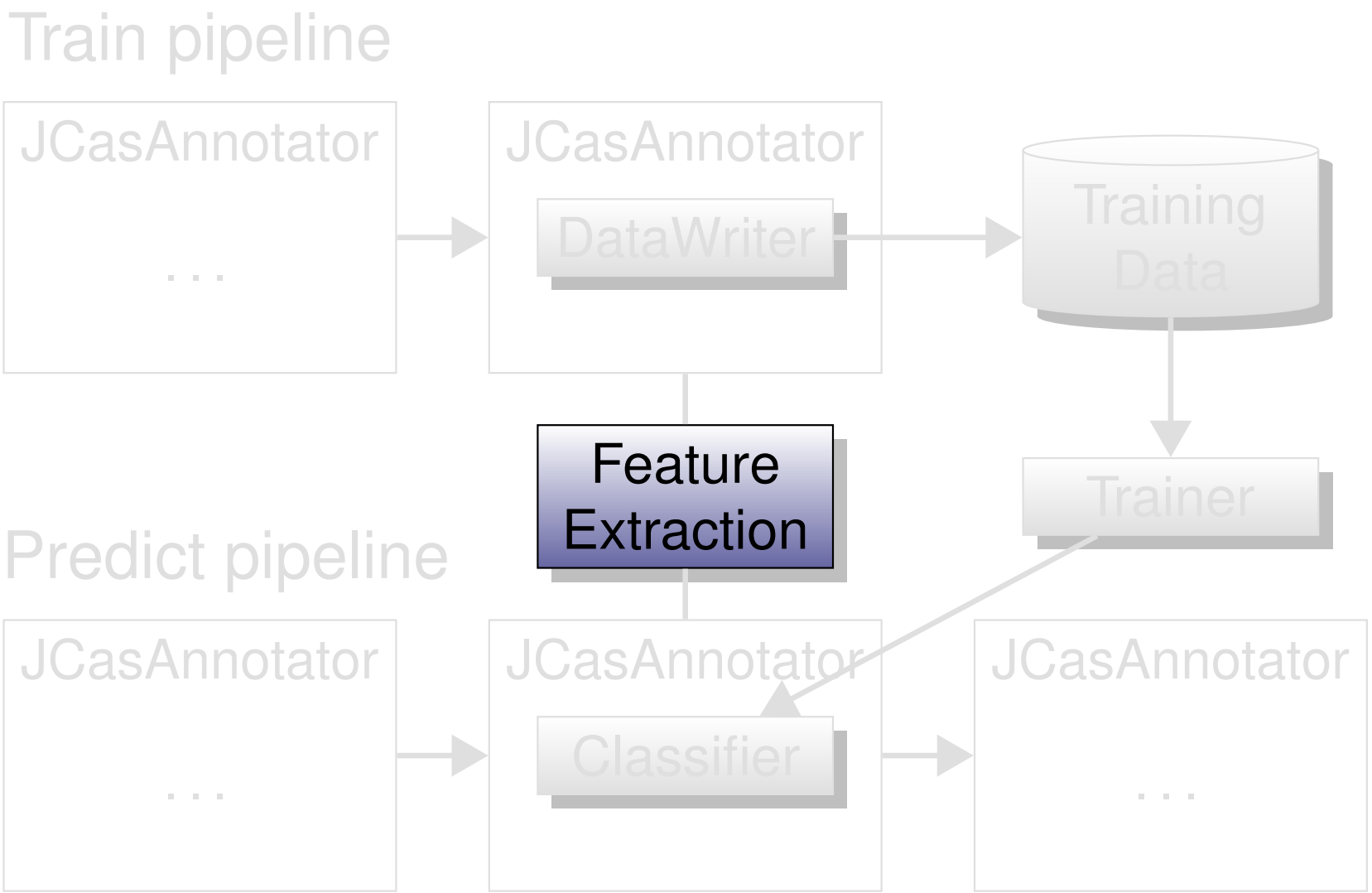
ClearTK Machine Learning Pipeline



ClearTK Machine Learning Pipeline



ClearTK Machine Learning Pipeline



Simple Feature Extraction

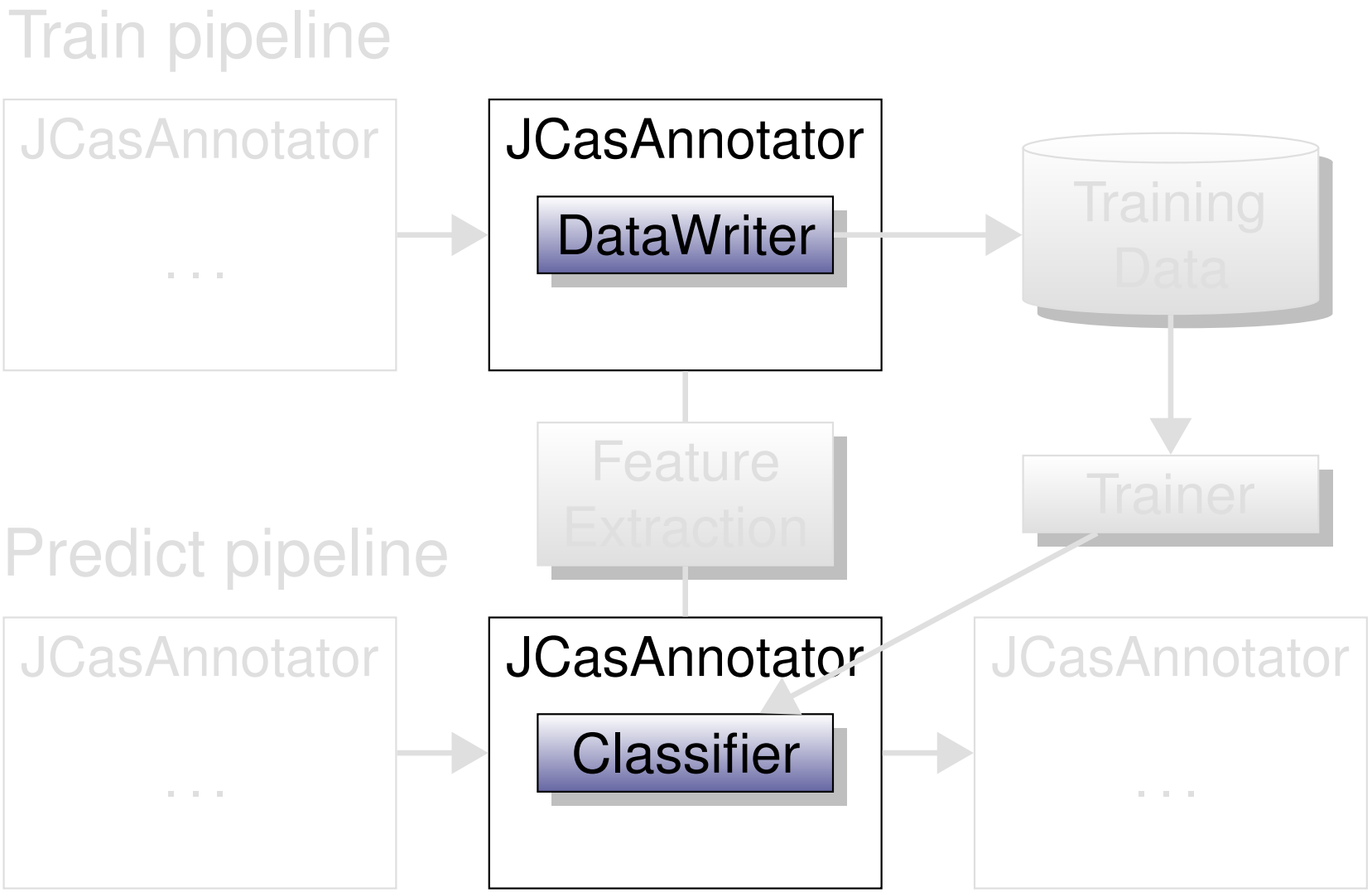
All code examples using ClearTK trunk, r3912, May 10, 2012

```
public void process(JCas jCas) {
  ...
  // extract an annotation from the CAS
  Token token = ...
  // create some features from it
  List<Feature> features = new ArrayList<Feature>();
  // create a feature from the text in the CAS
  int length = token.getCoveredText().length();
  features.add(new Feature("length", length));
  // create a feature from an annotation feature in the CAS
  String pos = token.getPartOfSpeech();
  features.add(new Feature("pos", pos));
  ...
}
```

Feature Extractor Library

```
public void process(JCas jCas) {
  ...
  // unicode patterns, e.g. "Az0" -> "LuLlNd"
  extractor = new CharacterCategoryPatternExtractor();
  ...
  // or features by navigating the CAS type system
  extractor = new TypePathExtractor(Token.class,"dep/head/pos");
  ...
  // or features from the surrounding context
  extractor = new ContextExtractor(Token.class,
    new CoveredTextExtractor(),
    new Preceding(2),
    new Ngram(new Following(3)));
  ...
  // apply the extractor to an annotation
  List<Feature> features = extractor.extract(annotation);
  ...
}
```

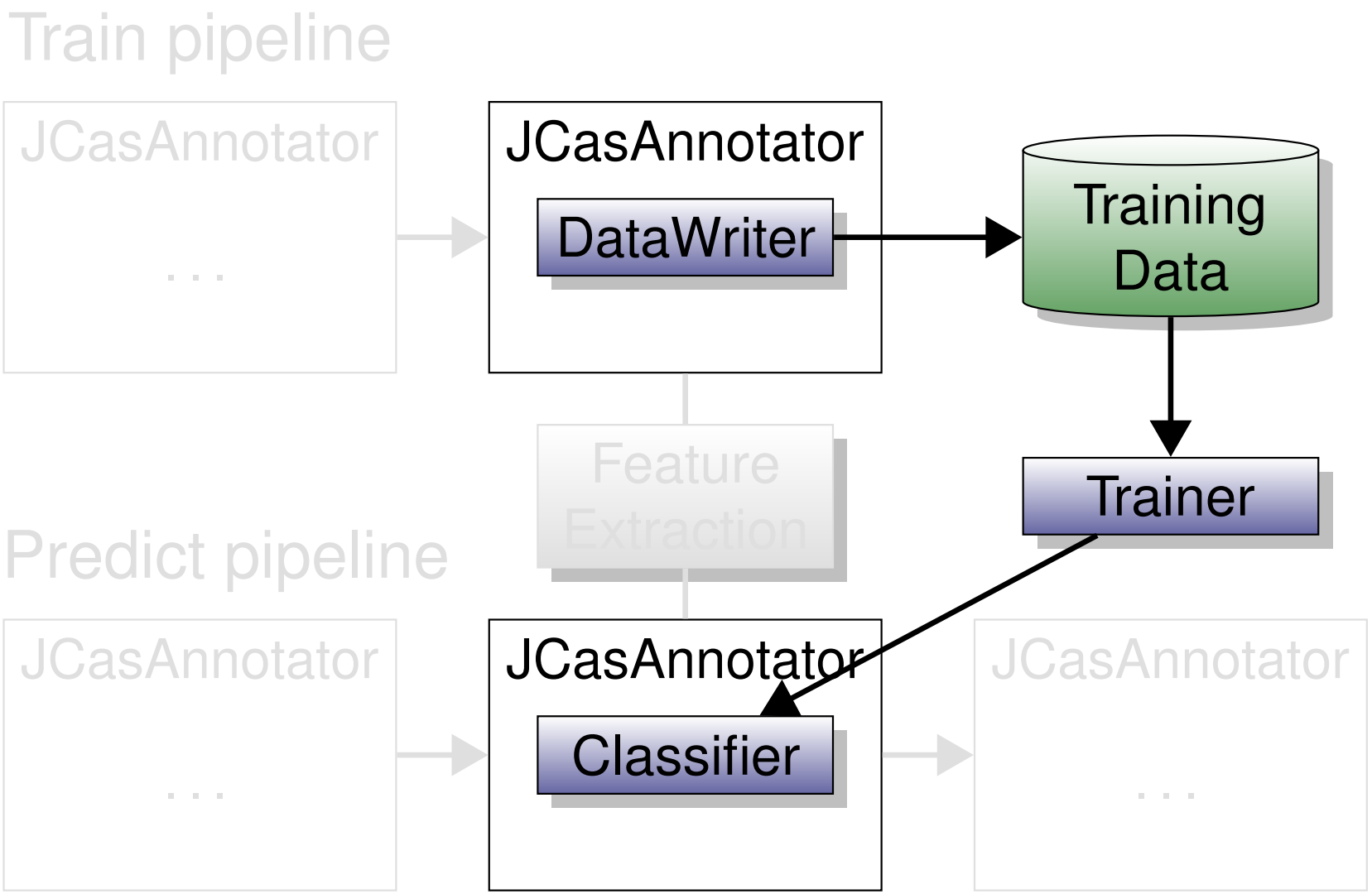
ClearTK Machine Learning Pipeline



Classifier Annotators (CleartkAnnotator)

```
public void process(JCas jCas) {
  ...
  // extract features
  List<Feature> features = ...
  // during training, create instances from CAS
  if (this.isTraining()) {
    String outcome = ... // e.g. token.getPOS()
    this.dataWriter.write(new Instance(outcome, features));
  }
  // during prediction, create CAS annotations
  else {
    String outcome = this.classifier.classify(features);
    ... // e.g. token.setPOS(outcome)
  }
  ...
}
```

ClearTK Machine Learning Pipeline



Train Pipeline with CleartkAnnotator

```
...
// create UIMA descriptor for train pipeline
AnalysisEngineFactory.createPrimitiveDescription(
   MyCleartkAnnotator.class,
   // specify type of classifier to write data for
   DefaultDataWriterFactory.PARAM_DATA_WRITER_CLASS_NAME,
   MultiClassLIBSVMDataWriter.class.getName(),
   // specify output directory for training data
   DirectoryDataWriterFactory.PARAM_OUTPUT_DIRECTORY,
   dir.getPath());
...
// run UIMA train pipeline
...
// train classifier and package into a jar file
JarClassifierBuilder.trainAndPackage(dir);
...
```

Predict Pipeline with CleartkAnnotator

```
...
// create UIMA descriptor for predict pipeline
AnalysisEngineFactory.createPrimitiveDescription(
   MyCleartkAnnotator.class,
   // specify where to load the classifier model from
   GenericJarClassifierFactory.PARAM_CLASSIFIER_JAR_PATH,
   new File(dir, "model.jar"));
...
// run UIMA predict pipeline
...
```

Summary

ClearTK machine learning framework:

- Feature extraction from CAS
- Common classifier interface to many libraries
- Training and loading classifiers from JARs

Many more features not covered in this talk, including:

- Sequence tagging (e.g. CRFs, Viterbi over k-best)
- Chunking (e.g. BIO tagging)
- Evaluation (e.g. cross-validation with UIMA pipelines)
- Regression and ranking (via SVM-light)
- Baselines (e.g. most frequent value, mean value)