// comments : knowledge approach might need revision

==========
Packages
==========

- WSDisambiguator
  + disambiguate
  + loadKnowledge
  + ...
  - WSDisambiguatorSupervised (mostly classification)
    + train
    + load
    + disambiguate
    + ...
  - WSDisambiguatorUnsupervised (mostly algorithmic and for baseline techniques)
    + disambiguate
    + ...
- WSDisambiguatorEvaluator
    + evaluate
    + getFMeasure
    + ...
+ WSDisambiguatorKnowledge

// other classes
- ResourceProvider
  + SenseProvider
  + KnowledgeProvider

==========
Classes
==========

WSDisambiguator : abstract class that has the common function of [disambiguate]

WSDisambiguatorUnsupervised : subclass of WSDisambiguator that is the model for implementations of unsupervised techniques

WSDisambiguatorSupervised : subclass of WSDisambiguator that is the model for implementations of supervised techniques

WSDisambiguatorEvaluator : will link to a WSDisambiguator object and disambiguate with input data for testing using [evaluate] method.

WSDisambiguatorKnowledge : will represent the extra knowledge (domain, context, etc.) required depending on the technique, this knowledge will have a different structure depending on the technique. Can be a key-value dictionary for example :
[
‘domain’ : ‘finance’
]

// other classes
ResourceProvider : can be either a sense inventory or knowledge provider (from local/remote resources)
SenseProvider: provides the sense inventory (word + definitions)
KnowledgeProvider: provides the knowledge required based on the type and source

==========
Signatures
==========
// returns the senses ordered by their score (best one first or only 1 in supervised case)
String[] disambiguate(String[] inputText,int inputWordposition);

// trains the classifier with the given parameters
WSDisambiguatorModel train(String language, String technique,
ObjectStream<WSDisambiguatorSample> sampleStream, TrainingParameters trainParams);

// will load an existing trained model
WSDisambiguatorModel load(String modelFileName);

// will load knowledge from a specific resource (needs discussion because highly depends on technique used)
WSDisambiguatorKnowledge loadKnowledge(String languageCode, knowledgeTypes, knowledgeSources);

// evaluator constructor with set its disambiguator
WSDisambiguatorEvaluator(WSDisambiguator disambiguator);

// will evaluate the performance of the disambiguator using the [disambiguate] function
void evaluate(ObjectStream<WSDisambiguatorSample> sampleStream);

==========
WSDisambiguatorSupervised.train : Train Model

```
================================================================
Charset charset = Charset.forName("UTF-8");
ObjectStream<String> lineStream = new PlainTextByLineStream(new
FileInputStream("en-wsd.train"), charset);
ObjectStream<WSDisambiguatorSample> sampleStream = new
WSDisambiguatorSampleDataStream(lineStream);

WSDisambiguatorModel model;

String technique = "technique1";
String languageCode = "en"

try {
  model = WSDisambiguatorSupervised.train(languageCode, technique, sampleStream,
TrainingParameters.defaultParams());
}
finally {
  sampleStream.close();
}

try {
  modelOut = new BufferedOutputStream(new FileOutputStream(modelFile));
  model.serialize(modelOut);
} finally {
  if (modelOut != null)
     modelOut.close();
}

================================================================
WSDisambiguatorSupervised.load : Load Model
================================================================
InputStream modelIn = new FileInputStream("en-wsd.bin");

try {
  WSDisambiguatorModel model = new WSDisambiguatorModel(modelIn);
}
catch (IOException e) {
  e.printStackTrace();
}
finally {
  if (modelIn != null) {
    try {
      modelIn.close();
```

```
    }
    catch (IOException e) {
    }
  }
}
```

```
================================================================
WSDisambiguatorSupervised.disambiguate : Disambiguate
================================================================

String[] inputText = "...";
int inputWordPosition = 10;

WSDisambiguator disambiguator = new WSDisambiguatorSupervised(model);
String[] orderedSenses = disambiguator.disambiguate(inputText, inputWordPosition);

================================================================
WSDisambiguatorUnsupervised.disambiguate : Disambiguate
================================================================

String inputText = "...";
int inputWordPosition = 10;

WSDisambiguator disambiguator = new
WSDisambiguatorUnsupervised(Params.defaultParams);
String[] orderedSenses = disambiguator.disambiguate(inputText, inputWordPosition);

================================================================
WSDisambiguatorUnsupervised.loadKnowledge : Load Knowledge
================================================================

// assuming this is somewhere before
WSDisambiguator disambiguator = new
WSDisambiguatorUnsupervised(params.defaultParams);

// assuming we have something like OPENNLP.CONSTANTS.TYPE1 ...
ArrayList<String> knowledgeTypes    = {OPENNLP.CONSTANTS.TYPE1,
OPENNLP.CONSTANTS.TYPE2} ; // "domain", "definitions"
ArrayList<String> knowledgeSources  = {OPENNLP.CONSTANTS.WORDNET}; // WordNet
String languageCode = "en";

WSDisambiguatorKnowledge knowledge = disambiguator.loadKnowledge(languageCode,
knowledgeSources, knowledgeTypes );
```

```
================================================================
WSDisambiguatorEvaluator.evaluate : Evaluate technique
================================================================
WSDisambiguator disambiguator =  new
WSDisambiguatorUnsupervised(params.defaultParams);
// or WSDisambiguator disambiguator = new WSDisambiguatorSupervised(model);

WSDisambiguatorEvaluator evaluator = new WSDisambiguatorEvaluator(disambiguator);
evaluator.evaluate(sampleStream);
FMeasure result = evaluator.getFMeasure();
System.out.println(result.toString());
================================================================
```