More (Advanced) JAPE

Module 1

June 2014

Outline

1. Debugging JAPE Grammars
2. Using Java in JAPE
    - Common idioms

Outline

1. Debugging JAPE Grammars

2. Using Java in JAPE
   - Common idioms

Debugging JAPE Grammars

- Read the error messages, they are helpful!
    - line numbers etc. refer to the original JAPE files
    - description usually highlights the exact problem

```
file:/home/gate/plugins/ANNIE/resources/NE/name.jape:
Encountered " <kleeneOp> "? "" at line 1580, column 10.
Was expecting one of:
    "\"" ...
    <ident> ...
    "|" ...
    "{" ...
    "(" ...
    ")" ...
```

Debugging JAPE Grammars

When trying to understand how annotations were created by a grammer try the new **enableDebugging** option:

- **addedByPR:** the name of the JAPE PR running the grammar that produced the annotation
- **addedByPhase:** the name of the phase (usually the filename) in which the annotation was created
- **addedByRule:** the name of the rule responsible for creating the annotation

Outline

1 Debugging JAPE Grammars

2 Using Java in JAPE
- Common idioms

Beyond Simple Actions

It's often useful to do more complex operations on the RHS than simply adding annotations, e.g.

- Set a new feature on one of the matched annotations
- Delete annotations from the input
- More complex feature value mappings, e.g. concatenate several LHS features to make one RHS one.
- Collect statistics, e.g. count the number of matched annotations and store the count as a document feature.

JAPE has no special syntax for these operations, but allows blocks of arbitrary Java code on the RHS.

Java on the RHS

```
Rule: HelloWorld
(
  {Token.string == "Hello"}
  {Token.string == "World"}
):hello
-->
{
  System.out.println("Hello world");
}
```

The RHS of a JAPE rule can have any number of `:bind.Type = {}` assignment expressions and blocks of Java code, separated by commas.

How JAPE Rules are Compiled

For each JAPE rule, GATE creates a Java class

```java
package japeactionclasses;
// various imports, see below

public class /* generated class name */
    implements RhsAction {
  public void doit(
      Document doc,
      Map<String, AnnotationSet> bindings,
      AnnotationSet inputAS,
      AnnotationSet outputAS,
      Ontology ontology) throws JapeException {
    // ...
  }
}
```

JAPE Action Classes

- Each block or assignment on the RHS becomes a block of Java code.
- These blocks are concatenated together to make the body of the `doit` method.
    - Local variables are local to each block, not shared.
- At runtime, whenever the rule matches, `doit` is called.

Java Block Parameters

The parameters available to Java RHS blocks are:

doc	The document currently being processed.
inputAS	The `AnnotationSet` specified by the `inputASName` runtime parameter to the JAPE transducer PR. Read or delete annotations from here.
outputAS	The `AnnotationSet` specified by the `outputASName` runtime parameter to the JAPE transducer PR. Create new annotations in here.
ontology	The ontology (if any) provided as a runtime parameter to the JAPE transducer PR.
bindings	The bindings map. . .

Bindings

- `bindings` is a `Map` from string to `AnnotationSet`
- Keys are labels from the LHS.
- Values are the annotations matched by the label.

```
(
  {Token.string == "University"}
  {Token.string == "of"}
  ({Lookup.minorType == city}):uniTown
):orgName
```

- `bindings.get("uniTown")` contains one annotation (the `Lookup`)
- `bindings.get("orgName")` contains three annotations (two `Token`s plus the `Lookup`)

Hands-on exercises

- The easiest way to experiment with JAPE is to use GATE Developer.
- The `hands-on` directory contains a number of sample JAPE files for you to modify, which will be described for each individual exercise.
- There is an `.xgapp` file for each exercise to load the right PRs and documents.
  - Good idea to *disable* session saving using Options → Configuration → Advanced (or GATE 8.0 → Preferences → Advanced on Mac OS X).

Exercise 1: A simple JAPE RHS

- Start GATE Developer.
- Load `hands-on/jape/exercise1.xgapp`
- This is the default ANNIE application with an additional JAPE transducer “exercise 1” at the end.
- This transducer loads the file `hands-on/jape/resources/simple.jape`, which contains a single simple JAPE rule.
- Modify the Java RHS block to print out the type and features of each annotation the rule matches. You need to right click the “Exercise 1 Transducer” and reinitialize after saving the `.jape` file.
- Test it by running the “Exercise 1” application.

Imports

- By default, every action class imports gate.*, java.io.*, java.util.*, gate.util.*, gate.jape.*, and gate.creole.ontology.*.
- So classes from these packages can be used unqualified in RHS blocks.
- You can add additional imports by putting an import block at the top of the JAPE file, before the `Phase:` line:

```
Imports: {
  import my.pkg.*;
  import static gate.Utils.*;
}
```

You can import any class available in the GATE core or in any loaded plugin. A useful class is gate.Utils, which provides static utility methods for common tasks that are frequently used in RHS Java code.

Named Java Blocks

```
-->
:uniTown{
  uniTownAnnots.iterator().next().getFeatures()
    .put("hasUniversity", Boolean.TRUE);
}
```

- You can label a Java block with a label from the LHS
- The block will only be called if there is at least one annotation bound to the label
- Within the Java block there is a variable *label*`Annots` referring to the `AnnotationSet` bound to the label
    - i.e. `AnnotationSet xyAnnots = bindings.get("xy")`

Exceptions

- Any `JapeException` or `RuntimeException` thrown by a Java RHS block will cause the JAPE Transducer PR to fail with an `ExecutionException`
- For non-fatal errors in a RHS block you can throw a `gate.jape.NonFatalJapeException`
- This will print debugging information (phase name, rule name, file and line number) but will not abort the transducer execution.
    - However it will interrupt this rule, i.e. if there is more than one block or assignment on the RHS, the ones after the **`throw`** will not run.

Returning from RHS blocks

- You can **`return`** from a Java RHS block, which prevents any later blocks or assignments for that rule from running, e.g.

```
-->
:uniTown{
  String townString = doc.getContent().getContent(
         uniTownAnnots.firstNode().getOffset(),
         uniTownAnnots.lastNode().getOffset())
       .toString();
  // don't add an annotation if this town has been seen before. If we
  // return, the UniversityTown annotation will not be created.
  if(!((Set)doc.getFeatures().get("knownTowns"))
       .add(townString)) return;
},
:uniTown.UniversityTown = {}
```

Common Idioms for Java RHS

Setting a new feature on one of the matched annotations

```
Rule: LcString
({Token}):tok
-->
:tok {
  for(Annotation a : tokAnnots) {
    // get the FeatureMap for the annotation
    FeatureMap fm = a.getFeatures();
    // get the “string” feature
    String str = (String)fm.get("string");
    // convert it to lower case and store
    fm.put("lcString", str.toLowerCase());
  }
}
```

Exercise 2: Modifying Existing Annotations

- Load `hands-on/jape/exercise2.xgapp`
- As before, this is ANNIE plus an extra transducer, this time loading `hands-on/jape/resources/general-pos.jape`.
- Modify the Java RHS block to add a `generalCategory` feature to the matched `Token` annotation holding the first two characters of the POS tag (the `category` feature).
- Remember to reinitialize the “Exercise 2 Transducer” after editing the JAPE file.
- Test it by running the “Exercise 2” application.

Common Idioms for Java RHS

Removing matched annotations from the input

```
Rule: Location
({Lookup.majorType = "location"}):loc
-->
:loc.Location = { kind = :loc.Lookup.minorType,
      rule = "Location"},
:loc {
  inputAS.removeAll(locAnnots);
}
```

This can be useful to stop later phases matching the same annotations again.

Common Idioms for Java RHS

Accessing the string covered by a match

```
Rule: Location
({Lookup.majorType = "location"}):loc
-->
:loc {
  try {
    String str = doc.getContent().getContent(
        locAnnots.firstNode().getOffset(),
        locAnnots.lastNode().getOffset())
      .toString();
  }
  catch(InvalidOffsetException e) {
    // can't happen, but won't compile without the catch
  }
}
```

Utility methods

- `gate.Utils` provides static utility methods to make common tasks easier
  - `http://gate.ac.uk/gate/doc/javadoc/gate/Utils.html`
- Add an `import static gate.Utils.*;` to your `Imports:` block to use them.
- Accessing the string becomes `stringFor(doc, locAnnots)`
- This is also useful for division of labour
  - Java programmer writes utility class
  - JAPE expert writes rules, importing utility methods

Example: start and end

To get the start and end offsets of an `Annotation`, `AnnotationSet` or `Document`.

```
Rule: NPTokens
({NounPhrase}):np
-->
:np {
  List<String> posTags = new ArrayList<String>();
  for(Annotation tok : inputAS.get("Token")
      .getContained(start(npAnnots), end(npAnnots))) {
    posTags.add(
        (String)tok.getFeatures().get("category"));
  }
  FeatureMap fm =
    npAnnots.iterator().next().getFeatures();
  fm.put("posTags", posTags);
  fm.put("numTokens", (long)posTags.size());
}
```

Exercise 3: Working with Contained Annotations

- Load `hands-on/jape/exercise3.xgapp`
- As before, this is ANNIE plus an extra transducer, this time loading `hands-on/jape/resources/exercise3-main.jape`.
- This is a multiphase grammar containing the `general-pos.jape` from exercise 2 plus `num-nouns.jape`.
- Modify the Java RHS block in `num-nouns.jape` to count the number of nouns in the matched `Sentence` and add this count as a feature on the sentence annotation.
- Remember to reinitialize the “Exercise 3 Transducer” after editing the JAPE file.
- Test it by running the “Exercise 3” application.

Passing state between rules

To pass state between rules, use document features:

```
Rule: Section
({SectionHeading}):sect
-->
:sect {
  doc.getFeatures().put("currentSection",
      stringFor(doc, sectAnnots));
}

Rule: Entity
({Entity}):ent
-->
:ent {
  entAnnots.iterator().next().getFeatures()
    .put("inSection",
         doc.getFeatures().get("currentSection"));
}
```

Passing state between rules

- Remember from yesterday - a `FeatureMap` can hold any Java object.
- So can pass complex structures between rules, not limited to simple strings.

Annotation Sets and Ordering

- An AnnotationSet is a set, so it is not ordered

```
Rule: SimpleNPRule1
(
  ({Token.generalCategory=="DT"})?
  ({Token.generalCategory=="JJ"})[0,4]
  ({Token.generalCategory=="NN"})+
):nnp
-->
:nnp {
  System.out.println("______________");
  System.out.println(stringFor(doc, nnpAnnots));
  System.out.println("The individual tokens:");

  for(Annotation tok : nnpAnnots) {
    System.out.println(stringFor(doc,tok));
  }
}
```

- The grammar for this example is in hands-on/jape/resources/match-nps.jape. To run the example yourself, load exercise2.xgapp in GATE Developer, load an extra JAPE Transducer PR, and give it as a parameter this grammar file. Finally, add the resulting new PR at the end of the Exercise 2 application and re-run it.

Annotation Sets and Ordering (Continued)

- Here is a sample output, if you execute this rule on our test document

```
waste management businesses
Now printing the matched individual tokens:
businesses
waste
management
```

- Instead, use from `gate.Utils` this method:

  `static List<Annotation> inDocumentOrder(AnnotationSet as)`, which returns a list containing the annotations in the given annotation set, in document order (i.e. increasing order of start offset).

- As an additional exercise, try instead to implement this functionality yourself, by modifying the RHS of the rule above and using the `OffsetComparator` from `gate.Utils`.