

William's Top 10 Things to Do With Minorthird

William W. Cohen
CALD

10. Play with the code

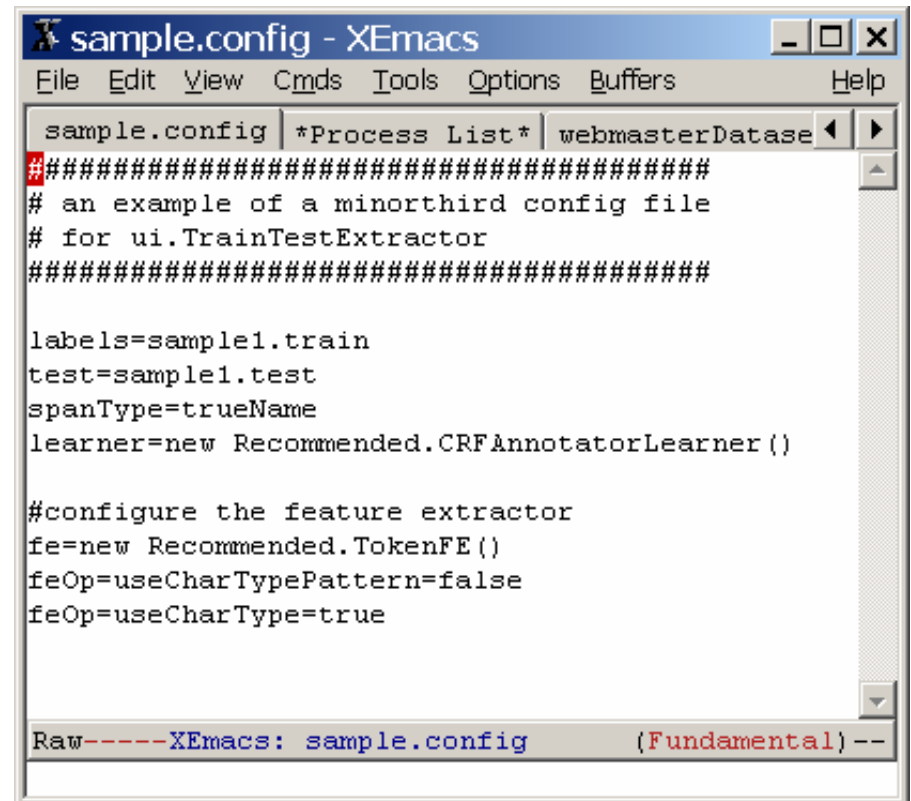
- The .jar file—all you need to use it is JRE 1.4+
 - <http://sourceforge.net/projects/minorthird>
- The latest source, with latest bug [fixes] and/or features:
 - <http://minorthird.sourceforge.net/IntroMinorthirdTutorial.doc> -> Part I
- The “launchpad”: `java edu.cmu.minorthird.Minorthird`
- The “user interface” package:
`java edu.cmu.minorthird.ui.Help`
- Every .ui program has a `-gui` option. Hit the ?’s and help buttons, and try the tutorials.
- The #1 most helpful feature of minorthird:
`cammie@cmu.edu`

9. Read the documentation

- The overview tutorial:
<http://minorthird.sourceforge.net/IntroMinorthirdTutorial.doc>
- The javadocs:
<http://minorthird.sourceforge.net/javadoc/edu/cmu/minorthird/package-summary.html>
- Sample code:
 - ~you/.../minorthird/demos
 - ~you/.../minorthird/apps/*

8. Use the command line

- Help available with
`java ...ui.XXX -help`
- Command-line options can be mixed with gui manipulations
- Any options can also be placed in a config file, eg.
`java ...ui.XXX -config sample.config`
- Command-line options and config files are for *reproducible* experiments



```
sample.config | *Process List* | webmasterDatase
#####
# an example of a minorthird config file
# for ui.TrainTestExtractor
#####

labels=sample1.train
test=sample1.test
spanType=trueName
learner=new Recommended.CRFAnnotatorLearner ()

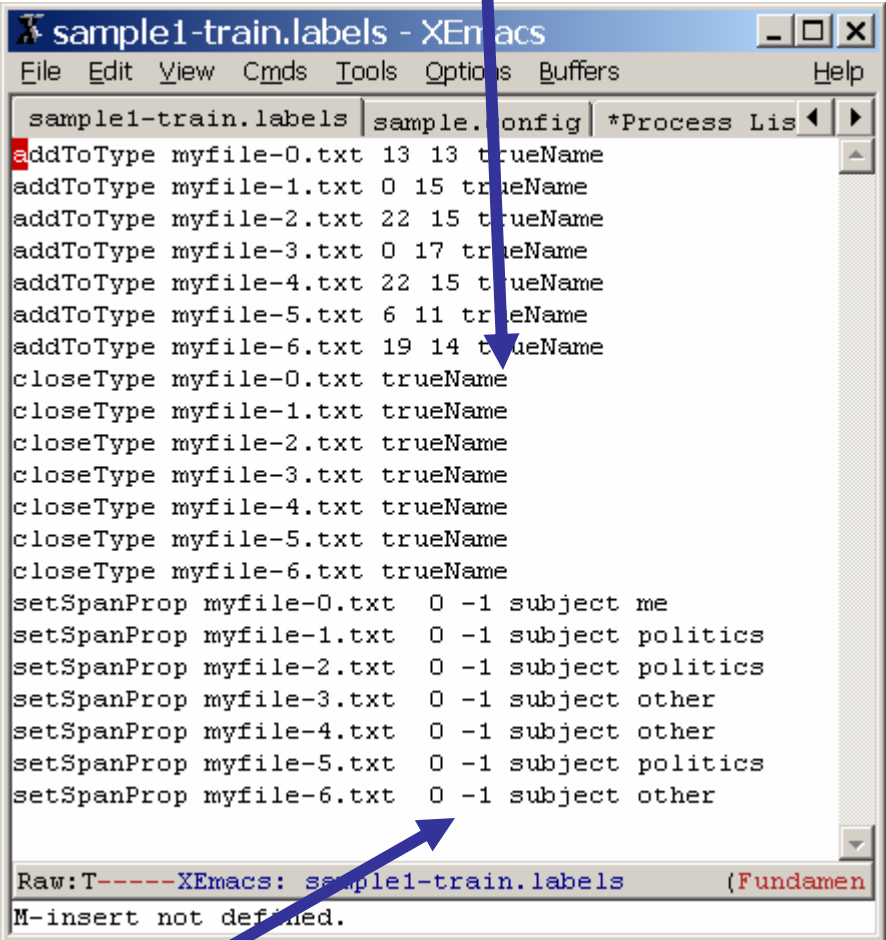
#configure the feature extractor
fe=new Recommended.TokenFE ()
feOp=useCharTypePattern=false
feOp=useCharType=true

Raw-----XEmacs: sample.config (Fundamental)---
```

7. Use your own dataset

“myfile-0.txt fully labels for trueName”

1. Make a directory **foo** of text files
2. Look at them with `...ui.ViewLabels`
3. Add annotations via “standoff annotation”: each line is
 - `addToType FILE LO LEN Type`
 - `closeType FILE Type`
4. `...ui.EditLabels` is an annotation tool & inline XML is also supported



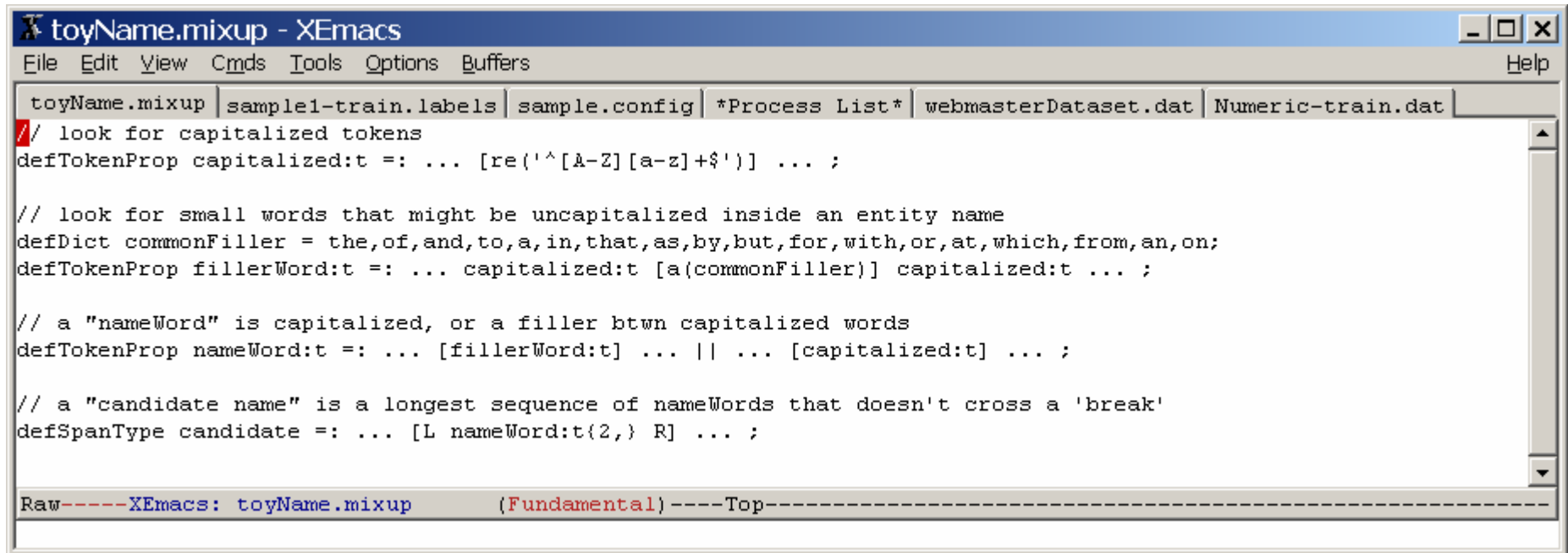
```
sample1-train.labels | sample. config | *Process Lis | Help
File Edit View Cmds Tools Options Buffers
addToType myfile-0.txt 13 13 trueName
addToType myfile-1.txt 0 15 trueName
addToType myfile-2.txt 22 15 trueName
addToType myfile-3.txt 0 17 trueName
addToType myfile-4.txt 22 15 trueName
addToType myfile-5.txt 6 11 trueName
addToType myfile-6.txt 19 14 trueName
closeType myfile-0.txt trueName
closeType myfile-1.txt trueName
closeType myfile-2.txt trueName
closeType myfile-3.txt trueName
closeType myfile-4.txt trueName
closeType myfile-5.txt trueName
closeType myfile-6.txt trueName
setSpanProp myfile-0.txt 0 -1 subject me
setSpanProp myfile-1.txt 0 -1 subject politics
setSpanProp myfile-2.txt 0 -1 subject politics
setSpanProp myfile-3.txt 0 -1 subject other
setSpanProp myfile-4.txt 0 -1 subject other
setSpanProp myfile-5.txt 0 -1 subject politics
setSpanProp myfile-6.txt 0 -1 subject other
Raw: T-----XEmacs: sample1-train.labels (Fundamen
M-insert not defined.
```

“-1” means to end of document

6. Pass data to your favorite learner

- ...ui.PreprocessTextForClassifier
- ...ui.PreprocessTextForExtractor
 - bug: both need options -saveAs foo.data from command line
- Output files:
 - lines in data file:
 - k textFileName class feature1=v1 feature2=v2 ...
 - a “*” indicates end of a sequence (= document) for extraction data
 - link file:
 - datafileName:lineNum textFileName LO LEN

5. Write a .mixup program



```
toyName.mixup - XEmacs
File Edit View Cmds Tools Options Buffers Help
toyName.mixup | sample1-train.labels | sample.config | *Process List* | webmasterDataset.dat | Numeric-train.dat
// look for capitalized tokens
defTokenProp capitalized:t =: ... [re('^[A-Z][a-z]+$')] ... ;

// look for small words that might be uncapitalized inside an entity name
defDict commonFiller = the,of,and,to,a,in,that,as,by,but,for,with,or,at,which,from,an,on;
defTokenProp fillerWord:t =: ... capitalized:t [a(commonFiller)] capitalized:t ... ;

// a "nameWord" is capitalized, or a filler btwn capitalized words
defTokenProp nameWord:t =: ... [fillerWord:t] ... || ... [capitalized:t] ... ;

// a "candidate name" is a longest sequence of nameWords that doesn't cross a 'break'
defSpanType candidate =: ... [L nameWord:t(2,) R] ... ;

Raw-----XEmacs: toyName.mixup (Fundamental) -----Top-----
```

... and run it (with `...ui.RunMixup`) or debug it (`...ui.DebugMixup`).

Using `-gui` and hitting “?” button brings you to a Mixup tutorial.

6. Reuse what you learn

- `ui.ApplyAnnotator` runs a learned annotator (produced by `ui.TrainClassifier` or `ui.TrainExtractor`) on a new dataset, and outputs the resulting annotation file as stand-off annotation (.labels file)

4. Write some features in mixup

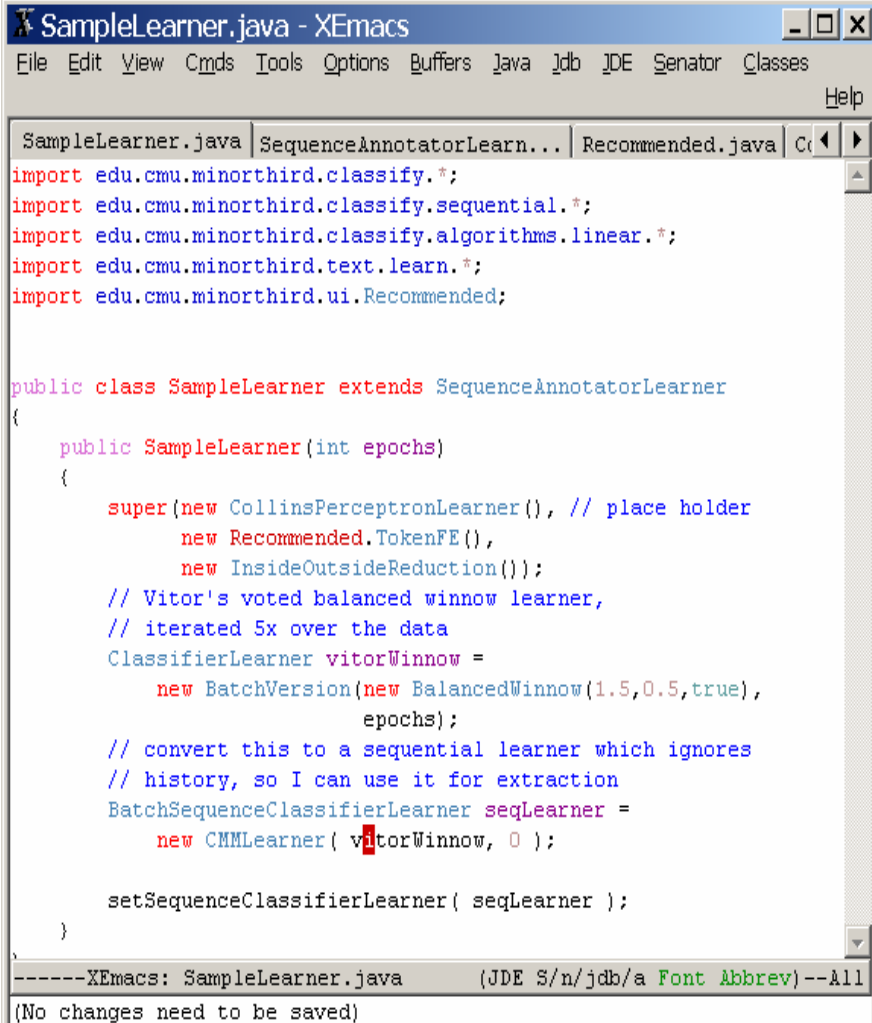
- By default, in extraction problems:
 - token properties (set with `defTokProp`) are exported to the learner as *features*
 - span properties and `spanTypes` (set with `defSpanProp`) are used as *training and test labels*—i.e., to define learning problems.
 - mixup is the recommended “feature engineering language” for minorthird.

3. Write a feature extractor in java

- A sample: `~you/.../minorthird/demos/MyFE.java`
- To use: compile and place the `.class` file in your classpath, then use `ui.TrainTestExtractor` options like `-fe "new MyFE()"`
- Simple things like a bag-of-words can be done with short sequences like:
 - `from(span).tokens().eq().emit(); // BOW`
 - `from(span).right().tokens(0).eq().emit(); // token to right`
- Complex things can be done by using the `minorthird.classify.*` API on the FE's protected variable **instance**.

2. Write your own learner

- Compile and use it the same way as for feature extractors: use `-learner` “new MyLearner()” option for `ui.TrainXY`
- No samples in demos/ but hitting the “?” in the GUI will show you the javadocs for existing learner classes.



```
SampleLearner.java - XEmacs
File Edit View Cmds Tools Options Buffers Java Jdb IDE Senator Classes Help

SampleLearner.java | SequenceAnnotatorLearn... | Recommended.java | C< | >

import edu.cmu.minorthird.classify.*;
import edu.cmu.minorthird.classify.sequential.*;
import edu.cmu.minorthird.classify.algorithms.linear.*;
import edu.cmu.minorthird.text.learn.*;
import edu.cmu.minorthird.ui.Recommended;

public class SampleLearner extends SequenceAnnotatorLearner
{
    public SampleLearner(int epochs)
    {
        super(new CollinsPerceptronLearner(), // place holder
              new Recommended.TokenFE(),
              new InsideOutsideReduction());
        // Vitor's voted balanced winnow learner,
        // iterated 5x over the data
        ClassifierLearner vitorWinnow =
            new BatchVersion(new BalancedWinnow(1.5,0.5,true),
                             epochs);

        // convert this to a sequential learner which ignores
        // history, so I can use it for extraction
        BatchSequenceClassifierLearner seqLearner =
            new CMMLearner( vitorWinnow, 0 );

        setSequenceClassifierLearner( seqLearner );
    }
}

-----XEmacs: SampleLearner.java (JDE S/n/jdb/a Font Abbrev)--All
(No changes need to be saved)
```

1. Be creative

- Look for useful main() programs:
 - `minorthird.text.SpanDifference` compares two label sets and prints precision/recall/F1
 - `minorthird.text.EncapsulatedAnnotator` lets you bundle a collection of mixup, dictionary, and class, files into a single self-contained file that can be run with any minorthird program.
- Talk to other systems: UIMA, Lucene?, Mallet?
- Play games with annotations:
 - `cat`, `awk`, `grep` are all valid tools
- Load any combination of documents & annotations
 - use API in `minorthird.text.TextBaseLoader`, `.TextLabelsLoader`
 - use beanshell script files as arguments to `-labels`
 - build a repository

But wait! you also get...

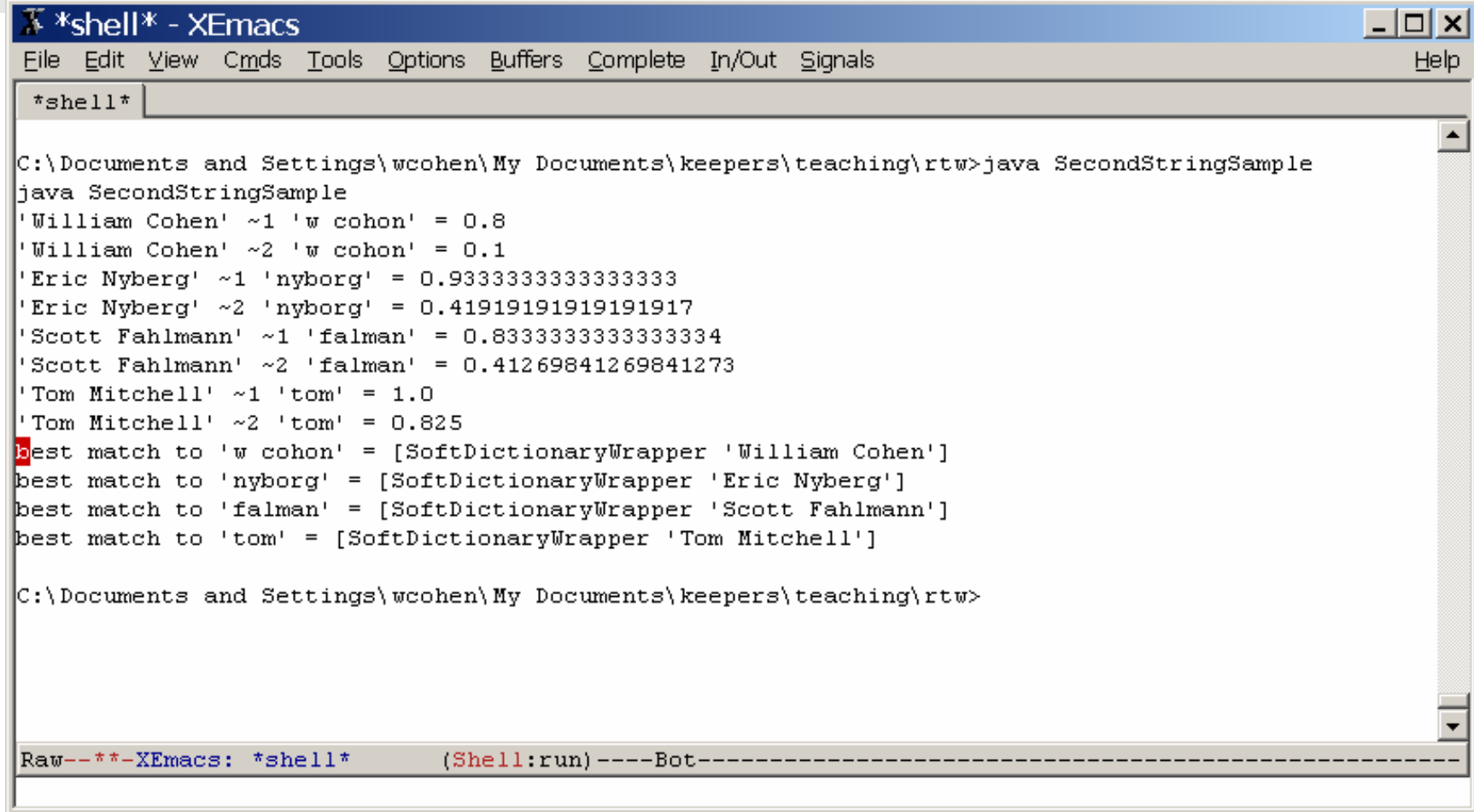
- SecondString is a separate Java SDK for string comparisons
- Source is at <http://secondstring.sourceforge.net/> and
- Javadocs are at <http://secondstring.sourceforge.net/javadoc/>
- Executables are included as a .jar library in Minorthird
 - so, you can call it directly if you have M3rd installed
- Some sample code is in the `com.wcohen.ss.expt` package, or for another example...

```
SecondStringSample.java - XEmacs
File Edit View Cmds Tools Options Buffers Java Jdb JDE Senator Classes Help
SecondStringSample.java | StringSimilarityAnnota... | AbstractStringDistance... | WinklerRescor...
import java.util.*;
import com.wcohen.ss.*;
import com.wcohen.ss.lookup.*;
import com.wcohen.ss.tokens.*;
import com.wcohen.ss.api.*;

public class SecondStringSample
{
    public static void main(String[] args)
    {
        String[] name = {"William Cohen", "Eric Nyberg", "Scott Fahlmann", "Tom Mitchell"};
        String[] misspelled = {"w cohon", "nyborg", "falman", "tom"};
        StringDistance distance1 = new MongeElkan();
        StringDistance distance2 = new JaroWinkler();
        for (int i=0; i<name.length; i++) {
            System.out.println("'" + name[i] + "' ~1 '" + misspelled[i] + "' = "
                + distance1.score(name[i], misspelled[i]));
            System.out.println("'" + name[i] + "' ~2 '" + misspelled[i] + "' = "
                + distance2.score(name[i], misspelled[i]));
        }

        SoftDictionary softDict = new SoftDictionary(); // use a default distance measure
        for (int i=0; i<name.length; i++) {
            softDict.put( name[i], null );
        }
        for (int i=0; i<misspelled.length; i++) {
            System.out.println("best match to '" + misspelled[i] + "' = "
                + softDict.lookup( misspelled[i]));
        }
    }
}

-----XEmacs: SecondStringSample.java (JDE S/n/jdb/a Font Isearch Abbrev) --All-- [M:main] ---
I-search:
```



```
*shell* - XEmacs
File Edit View Cmds Tools Options Buffers Complete In/Out Signals Help

*shell*

C:\Documents and Settings\wcohen\My Documents\keepers\teaching\rtw>java SecondStringSample
java SecondStringSample
'William Cohen' ~1 'w cohon' = 0.8
'William Cohen' ~2 'w cohon' = 0.1
'Eric Nyberg' ~1 'nyborg' = 0.9333333333333333
'Eric Nyberg' ~2 'nyborg' = 0.41919191919191917
'Scott Fahlmann' ~1 'falman' = 0.8333333333333334
'Scott Fahlmann' ~2 'falman' = 0.41269841269841273
'Tom Mitchell' ~1 'tom' = 1.0
'Tom Mitchell' ~2 'tom' = 0.825
best match to 'w cohon' = [SoftDictionaryWrapper 'William Cohen']
best match to 'nyborg' = [SoftDictionaryWrapper 'Eric Nyberg']
best match to 'falman' = [SoftDictionaryWrapper 'Scott Fahlmann']
best match to 'tom' = [SoftDictionaryWrapper 'Tom Mitchell']

C:\Documents and Settings\wcohen\My Documents\keepers\teaching\rtw>

Raw--**--XEmacs: *shell* (Shell:run)----Bot-----
```

More on secondstring...

- A distance may need to *prepare* a string for lookup (eg tokenize it)—and there's access to the “prepared” form, for efficiency
 - Prepared form is a *StringWrapper*
- A distance (eg TFIDF) might need to *compute statistics* over a set of strings, or might need to be trained from matching pairs. A *StringDistanceLearner* has various *train()* functions, and returns a trained *StringDistance*.
 - Most non-adaptive *StringDistance* classes also implement *StringDistanceLearner*