ShaC User's Guide

Matt Voss

September 21, 2005

1 Overview

This document is a user's guide for a Prolog program called A Shallow Syntactic Complexity Analyzer (ShaC) [1]. In general, ShaC takes a text as input and returns a syntactic complexity score for the text. The user is referred to [1] for a detailed description of what the program does. The following lists and describes predicates defined by that program. Note that there are many different ways in which one might want to interact with the core of this software. These predicates only give a few of the possibilities. The user knowledgeable in Prolog is encouraged to extend these predicates as necessary for individual projects.

2 Getting Started

All of the relevant files should be stored in `shac.ZIP`. Extract all files into the same directory. Consult `shac.pl` to begin using the software in the ways outlined below.

3 Templates

Many of the predicates in the sections below take as input a file listing the appropriate templates and each template's corresponding D-Level. For example, the template for a relative clause modifying the object of the verb is as follows:

```
template(3,[-,vpsed,n,comp]).
```

This predicate indicates that the template corresponds to a D-Level of 3 and that the template itself is `[-,vpsed,n,comp]`.

Bundled with the software is a file called 'templates.txt', which includes all the templates used during the creation of the program. The user may create other files including other templates for their personal use.

4 Files

Some of the predicates outlined below will rank all of the files in a list of files. These predicates assume the file listing all of the files to be analyzed by ShaC is a series of `file_loc/3` facts. For example:

```
file_loc(1,1,'filename1.txt').
file_loc(2,0,'filename2.txt').
file_loc(3,1,'filename3.txt').
```

The first argument is the file number, arbitrarily set by the user. The second argument indicates that the file should be read with a '1' and that it should not be read with a '0'. This allows the user to specify whether certain files should be skipped, a feature useful when dealing with large batches of files. The third argument lists the name of the file to be processed.

The file containing the listing of `file_loc/3` predicates should also include a `file/1`, whose argument is the number of `file_loc/3` listings in the file. For example, if you wanted to process 30 files, you would have 30 `file_loc/3` predicates, and in the file the fact

```
files(30).
```

5 Predicates

This section is divided into two parts. The first describes predicates used to test the validity of the program. The second section describes predicates used to rank and evaluate texts according the scoring system outlined in [1].

5.1 Test Predicates

The following predicates all read a text one line at a time, and tally how many of each template can be found in each line. I have assumed in writing them that the input text is formatted to have one sentence per line.

rank_a_sentence(+String,+TemplateFile)

This predicate takes as input a sentence, as a list of atoms, and a file with a listing of templates. It ranks the sentence according to the given templates and prints out the results as follows:

```
?- rank_a_sentence([the,dog,that,the,boy,liked,jumped,over,the,moon],'templates.txt').
sentence: [the, dog, that, the, boy, liked, jumped, over, the, moon]
phrase: [[the], dog, that, [the], boy, liked] count: 1
level: 3
template: [-, n, comp, n, vpsed]
line total score: 3
score level: 3 score: 1
```

```
Yes
```

The predicate prints several useful pieces of information: the sentence analyzed in list form, each phrase the program found as a list of atoms with optional words also in brackets, the D-Level of the phrase, the template used, the total score for the line, and a list of the D-Level followed by the number of phrases found at that level.

rank_sentences(+File,+Templates)

This predicate ranks all of the sentences in the file `File` according to the templates in `Templates` and prints the results to the screen. For example:

```
?- rank_sentences('testfiles//abcnews.txt','templates.txt').
% templates.txt compiled 0.00 sec, 2,336 bytes
sentence: [we, are, living, in, a, time, when, the, other, side, doesnt, want, us, to, s
phrase: [want, us, to, see] count: 1
level: 4
template: [-, vnfc, n, to, vpl]
phrase: [when] count: 1
level: 5
template: [-, subconj]
line total score: 9
score level: 4 score: 1
score level: 5 score: 1
sentence: [facts, are, inconvenient, facts, about, global, warming, facts, about, mercur
phrase: [said] count: 1
level: 3
template: [-, compv]
line total score: 3
score level: 3 score: 1
sentence: [the, former, first, lady, spoke, monday, at, a, new, york, women, for, hillar
line total score: 0
sentence: [she, leads, potential, gop, senate, opponents, to, in, recent, polls]
line total score: 0
sentence: [there, has, never, been, an, administration, i, dont, believe, in, our, histo
phrase: [power, to, further] count: 1
level: 1
template: [-, vpsed, to, vpl]
phrase: [believe] count: 1
phrase: [said] count: 2
level: 3
template: [-, compv]
line total score: 7
score level: 1 score: 1
score level: 3 score: 2
```

```
sentence: [a, spokeswoman, for, the, republican, national, committee, compared, clinton,
line total score: 0
sentence: [it, s, too, bad, that, new, york, s, senator, is, now, taking, her, cues, fro
phrase: [said] count: 1
level: 3
template: [-, compv]
line total score: 3
score level: 3 score: 1
sentence: [clinton, and, her, aides, maintain, that, her, focus, is, on, winning, a, sec
phrase: [in, [not], chasing] count: 1
level: 5
template: [-, ping, ving]
line total score: 5
score level: 5 score: 1
sentence: [but, republicans, say, her, sights, clearly, are, on, the, presidency]
phrase: [say] count: 1
level: 3
template: [-, compv]
line total score: 3
score level: 3 score: 1
```

rank_sentence_file(+File,+Templates,+OutFile)

This predicate ranks all of the sentences in `File` according to the templates in the file `Templates`, and writes the result to the file `OutFile`. It calls `rank_sentences/2`, described above.

```
?- rank_sentences('testfiles//abcnews.txt','templates.txt','example1.txt').

Yes
```

rank_sentences_files(+Files,+Templates,+OutFile)

This predicate ranks all of the sentences in a list of files specified by the file `Files` according to the templates found in the file `Templates`, and puts the result in the file `OutFile`. See Section 4 for an overview of the format `Files` must be in. For example:

```
?- rank_sentences_files('testers2.txt','templates.txt','example2.txt').
% testers2.txt compiled 0.01 sec, 4,220 bytes
% templates.txt compiled 0.01 sec, 2,304 bytes

Yes
```

5.2 Scoring Predicates

rank_single_level(+File,+Template,+Count)

This predicate prints out all occurences of the template `Template` in file `File`

```
?- rank_single_level('testfiles//abcnews.txt',[-,compv],X).
phrase: [said] count: 1
phrase: [believe] count: 2
phrase: [said] count: 3
phrase: [said] count: 4
phrase: [say] count: 5

X = 5

Yes
```

rank_atoms(+Sentence,+Template,-Score)

Ranks all of the atoms in the list of atoms `Sentence` according to the template `Template`. It returns either a 1 or 0 depending on success or failure, and prints out the phrase that matches the template, as follows:

```
?- rank_atoms([the,boy,that,the,girl,saw],[-,n,comp,n,vpsed],X).
phrase: [[the], boy, that, [the], girl, saw] count: 1

X = 1 ;

No
```

rank_files_to_file(+Files,+Templates,+OutFile)

Ranks the files in `Files` according to the templates in `Templates`, and puts the result in `OutFile`.

```
?- rank_files_to_file('testers2.txt','templates.txt','example3.txt').

Yes
```

rank_from_templates_score_list(+FileNum,+Files,+Templates,-Score)

Ranks file number `FileNum` from the file listing in `Files` according to the templates listed in the file `Templates`, returns a score for the text, and prints out a full listing of each phrase it found for each template. For example:

```
?- rank_from_templates_score_list(6,'testers2.txt','templates.txt',Score).
% testers2.txt compiled 0.00 sec, 4,188 bytes
level: 1
template: [-, vpsed, to, vpl]
```

```
phrase: [power, to, further] count: 1
level: 1
template: [-, vnfc, ving]
level: 3
template: [-, pnd, nom]
level: 3
template: [-, n, comp, vpsed]
level: 3
template: [-, n, comp, n, vpsed]
level: 3
template: [-, vpsed, comp, n, v]
level: 3
template: [-, compv]
phrase: [said] count: 1
phrase: [believe] count: 2
phrase: [said] count: 3
phrase: [said] count: 4
phrase: [say] count: 5
level: 3
template: [-, vpsed, n, comp]
level: 3
template: [-, it, bev, ving, comp2]
level: 3
template: [-, it, bev, vpsed, comp2]
level: 3
template: [-, it, bev, adj, comp2]
level: 4
template: [-, vnfc, n, to, vpl]
phrase: [want, us, to, see] count: 1
level: 4
template: [-, vnfc, nacc, vppl]
level: 4
template: [-, than]
level: 4
template: [-, as, adj, as]
level: 5
template: [-, subconj]
phrase: [when] count: 1
level: 5
template: [-, ping, ving]
phrase: [in, [not], chasing] count: 1
score level 1: 0.00520833
score level 1: 0
score level 3: 0
score level 3: 0
score level 3: 0
```

```
score level 3: 0
score level 3: 0.078125
score level 3: 0
score level 3: 0
score level 3: 0
score level 3: 0
score level 4: 0.0208333
score level 4: 0
score level 4: 0
score level 4: 0
score level 5: 0.0260417
score level 5: 0.0260417

Score = 0.15625 ;

No
```

rank_from_templates(+FileNum,+Files,+Templates,-Score)

This predicate is similar to `rank_from_templates_score_list/4` but does not print out the score at each level. For example:

```
?- rank_from_templates(6,'testers2.txt','templates.txt',Score).
level: 1
template: [-, vpsed, to, vpl]
phrase: [power, to, further] count: 1
level: 1
template: [-, vnfc, ving]
level: 3
template: [-, pnd, nom]
level: 3
template: [-, n, comp, vpsed]
level: 3
template: [-, n, comp, n, vpsed]
level: 3
template: [-, vpsed, comp, n, v]
level: 3
template: [-, compv]
phrase: [said] count: 1
phrase: [believe] count: 2
phrase: [said] count: 3
phrase: [said] count: 4
phrase: [say] count: 5
level: 3
template: [-, vpsed, n, comp]
level: 3
```

```
template: [-, it, bev, ving, comp2]
level: 3
template: [-, it, bev, vpsed, comp2]
level: 3
template: [-, it, bev, adj, comp2]
level: 4
template: [-, vnfc, n, to, vpl]
phrase: [want, us, to, see] count: 1
level: 4
template: [-, vnfc, nacc, vppl]
level: 4
template: [-, than]
level: 4
template: [-, as, adj, as]
level: 5
template: [-, subconj]
phrase: [when] count: 1
level: 5
template: [-, ping, ving]
phrase: [in, [not], chasing] count: 1

Score = 0.15625 ;

No
```

rank_from_templates(+FileName,+Templates,Score)

This predicate is similar to `rank_from_templates/4` described above, but instead takes the file name `FileName` directly as input. For example:

```
?- rank_from_templates('testfiles//abcnews.txt','templates.txt',Score).
level: 1
template: [-, vpsed, to, vpl]
phrase: [power, to, further] count: 1
level: 1
template: [-, vnfc, ving]
level: 3
template: [-, pnd, nom]
level: 3
template: [-, n, comp, vpsed]
level: 3
template: [-, n, comp, n, vpsed]
level: 3
template: [-, vpsed, comp, n, v]
level: 3
template: [-, compv]
```

```
phrase: [said] count: 1
phrase: [believe] count: 2
phrase: [said] count: 3
phrase: [said] count: 4
phrase: [say] count: 5
level: 3
template: [-, vpsed, n, comp]
level: 3
template: [-, it, bev, ving, comp2]
level: 3
template: [-, it, bev, vpsed, comp2]
level: 3
template: [-, it, bev, adj, comp2]
level: 4
template: [-, vnfc, n, to, vpl]
phrase: [want, us, to, see] count: 1
level: 4
template: [-, vnfc, nacc, vppl]
level: 4
template: [-, than]
level: 4
template: [-, as, adj, as]
level: 5
template: [-, subconj]
phrase: [when] count: 1
level: 5
template: [-, ping, ving]
phrase: [in, [not], chasing] count: 1

Score = 0.15625 ;

No
```

rank_files_sas_scores(+Files,+Templates,+OutFile)

Ranks the files in `Files` according to the templates in `Templates` and puts the results in `OutFile`. Output is formatted for SAS. It displays the file name and the overall score for each text separated by a space. For example, the following call

```
?-rank_files_sas_scores('testers2.txt','templates.txt','example4.txt').
% templates.txt compiled 0.01 sec, 2,336 bytes
% testers2.txt compiled 0.01 sec, 4,188 bytes

Yes
```

produces the output

```
testfiles\abcnews.txt 0.15625
testfiles\gray.txt 0.115578
```

in the file 'example4.txt'.

References

[1] Voss, M. (2005) *Determining Syntactic Complexity Using Very Shallow Parsing*. Master's Thesis, University of Georgia.