Automatic PCFG generator

Todd Layton, Anastasia Uryasheva

December 11, 2012

1 INTRODUCTION

Here will be an introduction

2 BACKGROUND

Here will be an article description

3 IMPLEMENTATION

The `pcfggen` module generates a PCFG through an iterative process of production rule creation and modification, which transforms an initial trivial grammar into a more generalized one.

3.1 INITIAL GRAMMAR

The program uses the input training corpus to create a simple grammar as a starting point for the rule modification. For each POS-tagged sentence in the training corpus, a corresponding production rule is added to the grammar. The left-hand side of this rule is `ROOT`, the right-hand side is the sequence of POS tags constituting that sentence, and the probability is 1 over the total size, in sentences, of the corpus.

3.2 ITERATION

Once the initial grammar is created, the program repeatedly looks for modifications to make in order to produce a grammar which is not so closely fitted to the training corpus. Generation of the PCFG is complete once there are no remaining modifications to be made. These

modifications consist of two types of grammar transformations: 2-gram expansion and rule joining.

3.2.1 2-GRAM EXPANSION

The program finds the most frequently-occurring symbol 2-gram in the training corpus which is not already the right-hand side of a production rule in the grammar. A new nonterminal symbol is created, and each instance of that 2-gram is replaced with that nonterminal. An additional production rule is added to the grammar, of which the left-hand side is the new nonterminal, the right-hand side is the 2-gram, and the probability is 1. This step does not change the functional structure of the grammar, but rather decomposed it into a larger number of simpler rules in order to enable rule joining.

3.2.2 RULE JOINING

Before each 2-gram expansion step, the program checks whether any multiple production rules in the corpus, with more than one symbol in the right-hand side, consists of the exact same right-hand side except for the symbols at one particular index (for example, the third symbol in each rule's right-hand side). If such rules are found, then a new nonterminal symbol is created, and the symbol at that index in each rule's right-hand side is replaced with that nonterminal. Then, for each of those rules, a new rule is added to the grammar, of which the left-hand side is the new nonterminal, the right-hand side is the symbol which that nonterminal replaced in that rule, and the probability is the relative frequency of the right-hand side symbol. Unlike 2-gram expansion, rule joining actually changes the functional structure of the grammar, allowing for symbol sequence which did not previously fit.

3.2.3 EXAMPLE

The effects of these two types of grammar modification are best illustrated with a simple example. Consider a toy corpus of three sentences, corresponding to three POS tag sequences:

(a) `A C B C`

(b) `X A C`

(c) `Y B C`

Given this corpus as the training input, the program would perform the following steps:

1. Create the initial grammar:
   (a) `ROOT →A C B C [1/3]`
   (b) `ROOT →X A C [1/3]`
   (c) `ROOT →B C Y [1/3]`

2. Check for rule joining, but no production rules are sufficiently similar.

3. Expand the 2-gram `A C`:

    (a) `ROOT →exp(A)(C)-0 B C [1/3]`

    (b) `ROOT →X exp(A)(C)-0 [1/3]`

    (c) `ROOT →B C Y [1/3]`

    (d) `exp(A)(C)-0 →A C [1]`

4. Check for rule joining, but no production rules are sufficiently similar.

5. Expand the 2-gram `B C`:

    (a) `ROOT →exp(A)(C)-0 exp(B)(C)-1 [1/3]`

    (b) `ROOT →X exp(A)(C)-0 [1/3]`

    (c) `ROOT →exp(B)(C)-0 Y [1/3]`

    (d) `exp(A)(C) →A C [1]`

    (e) `exp(B)(C) →B C [1]`

6. Join `A C` and `B C`:

    (a) `ROOT →exp(A)(C) exp(B)(C)-1 [1/3]`

    (b) `ROOT →X exp(A)(C)-0 [1/3]`

    (c) `ROOT →exp(B)(C)-1 Y [1/3]`

    (d) `exp(A)(C)-0 →join(A)(B)-2 C [1]`

    (e) `exp(B)(C)-1 →join(A)(B)-2 C [1]`

    (f) `join(A)(B)-2 →A [1/2]`

    (g) `join(A)(B)-2 →B [1/2]`

7. Check for rule joining, but no production rules are sufficiently similar.

8. Check for 2-gram expansion, but no unexpanded 2-grams remain.

9. Return the grammar.

4 RESULTS

Here will be a table with results and results' description

5 CONCLUSION

Here will be a conclusion