The Cloze Project
For Creating and Integrating Cloze Tests

Catherine E. Jones
catjones@holly.colostate.edu

Distributed Mentor Program Summer Research Project
Mentor: Professor Susan W. McRoy
Natural Language and Knowledge Representation Research Group
Electrical Engineering and Computer Science Department
University of Wisconsin-Milwaukee

1 Overview

The Cloze Project is an application designed to help developers design and integrate a Cloze Test into applications. It consists of two parts: the user interface to test and create a Cloze Test, and the Cloze Package to allow developers to integrate the test into an application. The Cloze Project is written in Java and uses the Java Swing package for the graphical user interface. It came about because there needed to be a simple, yet robust way of adding a reading comprehension test to applications. A Cloze Test is simple to implement on its own, but the Cloze Project has modified and expanded the original Cloze Test to allow for more flexibility and control.

1.1 Cloze Test

A Cloze Test is a standard reading comprehension test. In general, the Cloze Test keeps the first sentence intact and then deletes every fifth word after the first sentence. The length of the text is normally between 250-300 words. Researchers have shown that there is not a significant difference in grades when partial answers are counted (e.g. correct part of speech), so a direct comparison when grading is acceptable. Also, passing is generally 60% or above.

2 User Interface

The user interface is intended to help designers test and create different Cloze Tests. It is designed to be intuitive by displaying all the options to the user in an easy to navigate style. Figure 2.1 shows a screen shot of the main frame of the Cloze Project. This is the first screen that pops up. The menu bar has three different menus to help guide the user in making a Cloze Test.

2.1 The Test Menu

The test menu is where the user decides the features for their Cloze Test. Figure 2.1 displays the test menu. There are three different tests implemented:

- Every X-This test deletes every xth word, where is x is given by the window: 5, 10 or 20
- Low Frequency-This test deletes one low frequency word in a given window size: 5, 10 or 20. To know if a word is low frequency or not, this test makes a

frequency count of all the words in the given text. It then compares that words' frequency to see if it is less than the median frequency count. If so, the word is low frequency. This test was designed so that smaller words like 'the' and 'and' were not heavily tested.

- Hand Pick Victims-This test allows to user to hand select which words they want deleted. This could be useful to make short fill-in-the-blank tests, or other tests, if the above two tests don't suit the users' needs.

The numbers below the tests are where the user can choose the desired window size from the choices 5, 10, or 20. These numbers are only used in the first two types of tests.

After the test features have been picked, there are two options. Show Modified Text shows the text with the words to be deleted delimited with the pound (#) sign. Pressing Generate will create an actual test with the words blanked out so the user can try to take the test.

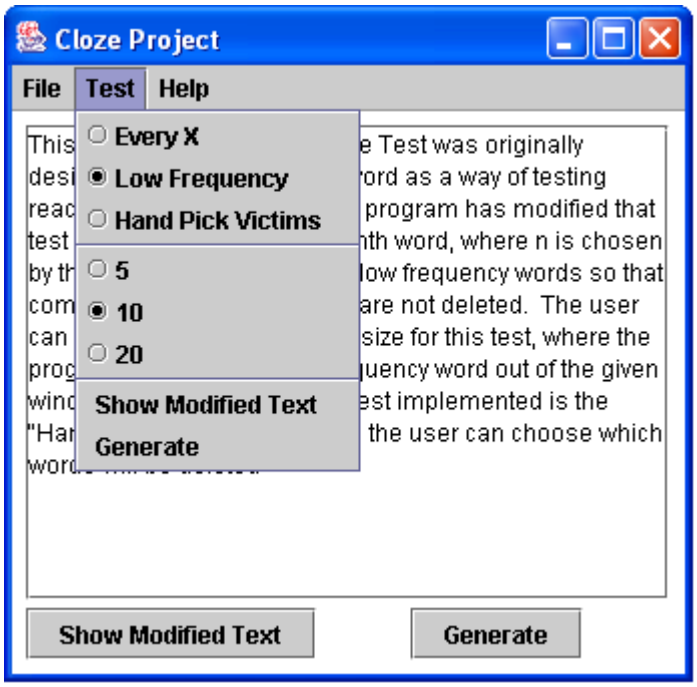

Figure 2.1: The main frame of the Cloze Project displaying the test menu

2.2 The File Menu

If the user has the text they want to turn into a test already stored in a text file, they can open it under the file menu by pressing 'open'. This extracts the text from the file and displays it in the text area. After a Cloze Test has been created, the user can save this test by pressing 'Save Test'. It can then be opened again by pressing 'Open Test' so the user can modify the test if needed. Pressing 'New' clears the text area and restores the default values for the test (Every X and 5).

2.3 The Help Menu

Under this menu the user can access the user manual if they need more instruction.

3 Cloze Package

There is also a package that developers can use to facilitate in adding a Cloze Test to an application. It is fairly simple to add a Cloze Test to an application. Mainly, the developer just needs the text that needs to be tested. An example bit of code is show in Example 3.1. This bit of code is all that is needed to integrate a Cloze Test. The developer can specify the type and the window if they need something other than the default Every X and 5. Removing the single quote is useful because a single quote can exist as part of a word as in can't and also to delimit words. Removing them assures that the test taker need not remember if there were quotes surrounding the word. The text should be set after all modifications (such as removing quotes) has been done. The next step, making the modified text, creates the delimited string which will be used in creating the actual test, which is shown when adminTest() is called.

Example 3.1: Java code to integrate a Cloze Test into an application

```
String str = "Once upon a time there was a girl.  This girl's name was Mary.
Mary liked sheep.  The end";
Cloze c = new Cloze();
c.setType("Low Frequency");
c.setNum("10");
str = str.removeChar(str, "/'"); //avoids confusion between can't and 'cat'
c.setText(str);
c.makeModifiedText();
c.adminTest();
```

4 Summary

The Cloze Project is a way for developers to test and create Cloze Tests. It provides an API for applications to integrate a Cloze Test. The user interface is a graphical environment that is intuitive to use which allows users to easily test and create a Cloze Test. For more information contact C. Jones at catjones@holly.colostate.edu or Professor S. McRoy at mcroy@tigger.cs.uwm.edu .