Laempel System
for
Intelligent Text
Processing

Getting Started

Installation

Create a new, empty directory called Laempel. Copy the file laempel.zip containing the entire system into Laempel. Unzip the file. At this time, the directory Laempel contains the file laempel.zip plus one file makefile.install, and it has four subdirectories: Code, Data, Makedata, and Manual. In turn, Code has four subdirectories: Create, Lbcode, Spell, and Syntax.

The subdirectories contain the following files.

Subdirectory Code/Create: create.zip of the C source code and main dictionary for creating index files used by the system.

Subdirectory Code/Lbcode: lbcode.zip of the C source code for logic reasoning.

SubdirectoryCode/Spell: spell.zip of the C source code for spell checking.

Subdirectory Code/Syntax: syntax.zip of the C source for syntax checking.

This work was funded in part by the Office of Naval Research under grant N00014-93-1-0096.

Version 7.0 12 September 2005

Subdirectory Data: data.zip of all data files used for spell and syntax checking.

Subdirectory Makedata: makedata.zip of test files and example files.

Subdirectory Manual: manual.zip of the manual in .ps and .pdf format. These files have been generated from Plain TEX files.

Unix/Linux Installation

It is assumed that the gcc compiler is available. The compilation and linking steps are accomplished by executing the single command `make -f makefile.install` in directory Laempel. The following programs are created.

Subdirectory Code/Create: lists all files, but does not compile or execute any programs.

Subdirectory Code/Lbcode: object files for linking with the spell and syntax checking routines.

Subdirectory Code/Spell: program laempel.spl for spell checking. The user may want to introduce a convenient alias. The use of the program is described in the next section.

Subdirectory Code/Syntax: program laempel.stx for syntax checking. The user may want to introduce a convenient alias. The use of the program is described in the next section.

The user should first try to run laempel.spl as described below in the section "The First Time." If execution of laempel.spl produces a run time error, then the index files connected with the main dictionary have the wrong format and must be replaced by correct index files. To create the correct index files, the user should go to the subdirectory Code/Create, open the file create.short.instruction using any text editor, and carry out the instructions given there. After that step, both laempel.spl and laempel.stx will function correctly.

CAUTION: Do not change any entry in any one of the data files of the subdirectory Data, except for a change of index files as described above.

All Other Installations

For all non-Unix/Linux installations, the following steps are required.

First, go into each one of the lowest subdirectories of Laempel and unzip the .zip file found there.

Second, in subdirectories Code/Lbcode, Code/Spell, and Code/Syntax of Laempel, use a suitable C/C++ compiler and carry out compilation and linking steps described in the makefile.install file of the subdirectories.

All C source code follows the ANSI standard and thus can be compiled by virtually any C/C++ compiler.

CAUTION: Compilation of C source code must first be done in subdirectory Code/Lbcode of Laempel, then in subdirectories Code/Spell and Code/Syntax.

The user should first try to run laempel.spl as described below in the section "The First Time." If execution of laempel.spl produces a run time error, then the index files connected with the main dictionary have the wrong format and must be replaced by correct index files. To create the correct index files, the user should go to the subdirectory Code/Create, open the file create.short.instruction using any text editor, and carry out the instructions given there. After that step, both laempel.spl and laempel.stx will function correctly.

CAUTION: Do not change any entry in any one of the data files of the subdirectory Data, except for a change of index files as described above.

Changing the Main Dictionary

The main dictionary word.dat in subdirectory Data can be expanded if so desired. This is a nontrivial process, since the user must define the syntactic roles the new words may take on. We omit the detailed and rather complicated discussion and instead describe a process that is much simpler, yet equally effective. We describe the steps using addition of the word "email". First, one finds a closely related word that syntactically can play the same roles as "email" and that occurs already in the dictionary. A good choice would be "mail." Second, one types the world "mail" into a file and applies laempel.spl to that file. That program inserts into the file words.usr an entry for "mail." One now creates another entry in words.usr that looks exactly like the one for "mail" except that the word "mail" is replaced by "email." That entry is inserted at the end of the file words.usr. From now on, laempel.spl and laempel.stx not only know the word email, but also know the syntactic roles that the word can take on.

Of course, laempel.spl can also learn words directly while processing text. The user simply directs laempel.spl to accept the unknown word. At that time, laempel.spl guesses how the word can occur syntactically. However, that guess may not be correct and thus may weaken the accuracy with which laempel.stx checks sentences for correct syntax.

The First Time

For the first-time user, here is an easy-to-follow recipe to get going.

1. Go to the directory containing the text files to be processed. Copy the file params.dat from the subdirectory Makedata into that directory.
2. Use any text editor to access the file params.dat, and modify that file as follows:

   Replace on the line
   `user history file = history.usr`
   the file name history.usr by another name if a file named history.usr exists already.

   Replace on the line
   `user dictionary = words.usr`
   the file name words.usr by another name if a file named words.usr exists already.

   Replace on the line
   `user dictionary backup = words.bak`
   the file name words.bak by another name if a file named words.bak exists already.

   Replace on the line
   `user phrase pattern = patterns.usr`
   the file name patterns.usr by another name if a file named patterns.usr exists already.

   Replace on the line
   `user phrase pattern backup = patterns.bak`
   the file name patterns.bak by another name if a file named patterns.bak exists already.

   Replace on the line
   `user suggested replacements = replace.usr`
   the file name replace.usr by another name if a file named replace.usr exists already.

Replace on the line
`file of words excluded by user = nono.usr`
the file name nono.usr by another name if a file named nono.usr exists already. The file nono.usr is controlled by the user. It contains excluded words, one word per line.

Replace on the line
`file name storage = filename.sav`
the file name filename.sav by another name if a file named filename.sav exists already.

Specify 'yes' or 'no' on the line
`accept all unknown words as correct ('yes' or 'no') = yes`

'yes' implies that the system accepts each word as correct. This option is useful for building the user dictionary words.usr from files that are known to be correct. 'no' implies that the system declares any word that is not contained in the main dictionary or in the user dictionary words.usr, to be incorrect.

Specify 'yes' or 'no' on the line
`find syntactical interpretations ('yes' or 'no') = yes`
'yes' implies that the system finds a syntactical interpretation for each sentence. 'no' implies that the system only identifies syntactical errors.

Specify 'int' or 'bat' on the line
`interactive or batch syntax checking('int'or'bat') = int`
'int' implies that the system does interactive syntax checking. 'bat' implies that the system does batch syntax checking.

Replace on the line
`directory of Laempel/Data = /home/Laempel/Data`
the path name /home/Laempel/Data by the full path name leading to Laempel/Data.

3. You are now ready for spell and syntax checking.

Spell Checking

1. Execute the program laempel.spl for spell checking.
2. The system asks permission to create the following files if they do not exist: user dictionary, the history file, file of user suggested replacements, and the file of excluded words. The file names are specified in params.dat. Before processing any text files, the system loads the main dictionary into memory. Due to the size of the dictionary, this step takes several seconds.
3. The following message and prompt appear:
   ```
   Name of text file
   Press Return for 'myfile.ext' (=most recent name)
   'q' to quit
   SPELLING >>
   ```
   Type the name of a text file to be processed and press Return, or press Return to process the most recent file.
4. If 'yes' has been specified in params.dat on the line
   `accept all unknown words as correct ('yes' or 'no') = no`
   then the system gives a warning message and asks for confirmation that the user indeed wants to accept all unknown words as correct.

   If the text contains misspelled words, and 'no' has been specified in params.dat on the line
   `accept all unknown words as correct ('yes' or 'no') = no`
   then the system locates misspelled words and proposes alternative words. The procedure is as follows.

   Each misspelled word and the text line containing it are displayed, followed by two or three suggested alternatives that allow the user to accept the word or to replace it by some other word. The options are numbered 1, 2, and 3. To accept one of the options, the user types in the option number and presses Return. To select option 1, the user may simply press Return. If none of the options apply, the user may type in the desired word, or specify the option '0 to have the word deleted from the text. If a typed-in word is not part of the dictionary, the system issues a warning.

Before the system implements the chosen alternative, it asks for confirmation that the choice is indeed the desired one. If the user has made a mistake, this is the time to correct the choice.

5. When spell checking of a text file has been completed, the system updates the user dictionary and user history file. `DO NOT ABORT` the system at that time, since any interruption at that point destroys the user dictionary and user history file. Also, the text file is updated if changes have been confirmed, and the original text file is saved with the extension .bak.

   When all files have been updated, the system displays
   ```
   Name of text file
   Press Return for 'myfile.ext' (=most recent name)
   'q' to quit
   SPELLING >>
   ```
   and is ready to process another text file. Type the name of a file, press Return to process the most recent file, or type 'q' to quit the Spell Checking Module.

CAUTION: It is highly recommended that the window displaying the revised text file be deleted. The reason is that any subsequent spell checking may change the text file. At that time, the stored text file may no longer agree with the text file displayed in the window.

Use of Text Files Known to be Correct

The directory Makefile contains a LaTeX file that is known to be correct. The user may apply the spell checking program laempel.spl to that file under the option
`accept all unknown words as correct ('yes' or 'no') = yes`
to create an initial words.usr file that contains frequently occurring LaTeX commands. That step can also be used in other settings where files with specialized terms are known to be correct. By spell checking these files with the above option, all such special words become part of the user dictionary and are known to the system from then on.

Exclusion of Words

The spell checker generally accepts both British and US spelling of words. For example, both "labelled" (British spelling) and "labeled" (US spelling) are accepted. If the user wants to exclude some of these words or, for that matter, any other words, he/she should enter them into the user-defined file nono.usr, one word per line. The directory Makedata contains an example nono.usr file that excludes a few words of British use. Each word of nono.usr is considered incorrect by the spell checker regardless of circumstance.

Syntax Checking

1. Execute the program laempel.stx for syntax checking.
2. If the file of user phrase patterns does not exist, the system asks permission to create that file using the file name specified in params.dat.

   The following message and prompt appear:
   ```
   Name of text file
   Press Return for 'myfile.ext' (=most recent name)
   'q' to quit
   SYNTAX >>
   ```
   Type the name of a text file to be processed and press Return, or press Return to process the most recent file. Make sure that the text file to be processed has already been spell checked.

   If 'int' has been specified in params.dat on the line
   `interactive or batch syntax checking ('int' or 'bat') = int`
   then go to Step 3.

   Alternately, if 'bat' has been specified in params.dat on the line
   `interactive or batch syntax checking ('int' or 'bat') = int`
   then go to Step 4.
3. (Interactive syntax checking) Each sentence with syntactical errors and the text line containing it are displayed, followed by an explanation of the error and the place where the error occurs. Type 'r' and press Return to reject the sentence if there is indeed an error. Type 'a' and press Return to accept the sentence if it is actually correct. Before the system rejects or accepts the sentence, it asks for confirmation of the decision.

   While the syntax checking is being performed, the user may edit the text file in a separate window, using any text editor. The changes so made by the user do not interfere with the syntax checking since the Syntax Checking Module does not read the text file. Go to Step 5.

4. (Batch syntax checking) Each sentence with syntactical errors, the text line containing it, an explanation of the error, and the place where the error occurs, are saved in a file. The name of that file is the name of the text file plus the extension .err.

   When syntax checking has been completed, the user may access the error file to make appropriate changes in the text file.

5. When syntax checking of a text file has been completed, the system updates the user dictionary and user phrase pattern file. `DO NOT ABORT` the system at that time, since any interruption at that point destroys the user dictionary and user phrase pattern file.

   When all files have been updated, the system displays
   ```
   Name of text file
   Press Return for 'myfile.ext' (=most recent name)
   'q' to quit
   SYNTAX >>
   ```
   and is ready to process another text file. Type the name of a file, press Return to process the most recent file, or type 'q' to quit the Syntax Checking Module.

CAUTION: If syntax checking initially was done in batch mode, any subsequent run of the Syntax Checking Module should be done interactively. The system then learns user preferences and exceptions concerning syntax, and adapts its reasoning accordingly.

6. Once the user has corrected the text file, the revised file should be saved.