Textchk

a simple text scanner that checks for common syntax and style rules

2001.08.29

Daniele Giacomini

Table of Contents

Introduction

This is the documentation for Textchk. I decided to write this simple program to help me to find my usual mistakes when I was writing an italian book about GNU/Linux and free software: Appunti Linux (`http://www.pluto.linux.it/ildp/appuntilinux/`).

I was convinced to translate this program into English and to make it as more generalized as possible, as before it was made only for my own formatting system (ALtools).

> I am sorry, but my English is very poor. Any comment and language correction to this manual is appreciated.

License

Textchk is released under the GNU General Public License.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Obtain Textchk

At the moment, the main distribution source for Textchk is the following URI:

`http://master.swlibero.org/~daniele/software/textchk/`

How to contact the author

Daniele Giacomini
Via Turati, 15
I-31100 Treviso
Italy

daniele @ swlibero.org

1 The problem to solve

Human writers make mistakes. With the help of a spell checker can be found only words wrongly spelled, but nothing more. Every one has it's own typical mistakes, that maybe can be found using simple regular expression.

Mistakes are not absolute; as languages are dynamic and every author may decide the style. Textchk helps with the definition of rules that define a kind of mistake. For example, '`\b[Tt]his *this\b`' is a regular expression that catch the use of the word "this" for two times (the first time can be capitalized), and this is presumably an error.

Error like these may be typical for one person and very unusual for the other. Textchk is made to let crate personalized rules, following the needs. These rules are mainly thought to be part of a particular documentation project; but can be defined also personal rules (valid for any personal documentation project) and also general rules to be extended system-wide.

2 Configuration

Configuration of Textchk is made of file that defines error rules (with exceptions) and special situation that are not to be considered mistakes for some reasons. The file that contains error and exception rules is organized with records like this:

'DBL____*error-rule*[____*explanation-text*]'

'ERR____*error-rule*[____*explanation-text*]'

'EXC____*exception-rule*'

Empty lines and lines that start with a '#' are ignored.

The four '_' are used to separate fields. The first one defines the type of record: 'DBL' means that the record describes a word repeated with no reason; 'ERR' means that the record describes an error; 'EXC' means that the record describe an exception for the previous error. The second filed is a regular expression that describe an error or an exception, depending on the first field. The third field is available to explain the error. An example may help:

```
ERR____\bI'm\b____I'm --> I am
EXC____\bI'm going\b
EXC____\bI'm very proud\b
```

In this case, it is considered an error to use 'I'm', because the author like more to expand it to 'I am'. The description to the error is very simple, 'I'm --> I am', but can be also more clear (something like 'I do not want things like "I'm"'). But this error has two exceptions: 'I'm going' and 'I'm very proud' are allowed.

When Textchk finds a correspondence with an error rule, it isolates the text around the error, exactly tree words before and three words after. Of course, there may be less of three words available. After that, the comparison with exceptions is made using this extracted text. This means that the following exception cannot be ever found, because there are four words after the text that is identified as an error.

```
ERR____\bI'm\b____I'm --> I am
# The following exception cannot be verified.
EXC____\bI'm very very very proud\b
```

Regular expressions that describe errors and exceptions should not include reference to the beginning and the end of a text line. That is: regular expression like '^...$' are not allowed.

The 'DBL' record describes a word what might appear double times, intended as an error. For example:

```
DBL____\w\w+____Doubles
EXC____\b[bB]ye\s+bye\b
```

In that case, any two or more alphanumeric characters, making a word, are located if written double time. Something like: "I need need money". The word "need" is written twice, and it is a mistake. As it can be seen, the exception showed inside the example means that the sequence "bye bye", or "Bye bye" must be allowed.

2.1 Configuration hierarchy

Textchk is thought to be used with configuration specific for every documentation project that any author can handle. Anyway, it is also possible to define a personal configuration and a system-wide configuration. Here are the configuration files for error and exceptions; at least one of these files is required:

1. '`./.textchk.rules`' is the current configuration, that is read before the other;
2. '`~/.textchk.rules`' is the personal configuration, that is read after the current one and before the system-wide configuration;
3. '`/etc/textchk.rules`' is the system-wide configuration, that is read after the others.

Generally it is better to avoid the use of a system-wide configuration. Anyway, if there is the need to override a system-wide rule, the same rule can be inserted inside the personal or current configuration file, followed with an exception with the same regular expression. That is; suppose that a system-wide rule is as it follows:

```
ERR____\bI'm\b____I'm --> I am
```

If you don't want to be bored with that, you can add this to your personal or current configuration:

```
# Override system-wide rule.
ERR____\bI'm\b
EXC____\bI'm\b
```

2.2 Special cases

Some times it is not convenient to define an exception rule for a particular error. Textchk generates a file containing the peaces of text containing the errors found. If some of these peaces of text are no mistakes, but you don't want to describe an exception to avoid this warning, you can copy them into '`./.textchk.special`' (there is no personal, nor system-wide one).

Suppose that you run Textchk and you obtain a report made of the following lines, because you decided that "I'm" is a mistake:

```
this is because I'm over the big
I'm out of control
I'm not going anywhere
```

Suppose that you don't want to be warned when the peace of text is '`I'm not going anywhere`'. Just put that line into the file '`./.textchk.special`', and you will not see this warning anymore.

```
I'm not going anywhere
```

Now should be clear that the file '`./.textchk.special`' is only for special exceptions: no regular expressions, but only pure text. Eventually, empty lines are ignored, but no comments are allowed.

3 Input for the analysis

Textchk read the input file line by line and the comparison with error rules is made inside the space of a single line. This way, the text file that is used as an input, should be transformed so that paragraphs are joined together; that is: every paragraph should stay on a single line.

This job is made by a front-end for man pages, HTML pages and Texinfo sources. For other sources, the text must be normalized as a simple text file with very long lines.

4 How to use Textchk

Textchk is made of one single executable: `textchk`.

textchk *option file-to-be-analyzed* [*report-file* [*diag-file*]] Command

The option defines the type of the file, '`--input-type=`*type*', so that it can be transformed before the real scan. Some key words are available:

- '`man`' means that this is a man page;
- '`html`' means that this is an HTML page;
- '`texinfo`', '`texi`' means that this is a Texinfo source;
- '`standard`' means that this is a normalized text file.

The second argument is the name of the file. The third argument can be the name of the report file (the one that store the peaces of text considered mistakes); if not given it is equal to '*file-to-be-analyzed*`.err`'. The fourth argument is the name for a diagnostic file, that contains all information of the scanning made, useful to understand where rules doesn't do what is expected. If this name is not given, it is equal to '*report-file*`.diag`' or '*file-to-be-analyzed*`.diag`'.

For example,

'`textchk --input-type=man bash.1`'

gives two files: '`bash.1.err`' and '`bash.1.diag`'.

4.1 How errors are shown

During its work, Textchk shows on screen what it finds, delimiting errors with '`>>`' and '`<<`'. For example, if we have the same old error rule:

```
ERR____\bI'm\b____I'm --> I am
EXC____\bI'm going\b
```

we can obtain warning like these:

```
I'm --> I am
to be here. >>I'm<< here today and
I'm --> I am
>>I'm<< not mad.
```

Inside the diagnostic report, all the process is shown:

```
??? to be here. >>I'm<< here today and
ERR \bI'm\b
!!! to be here. >>I'm<< here today and

??? I know, >>I'm<< going to be
ERR \bI'm\b
EXC \bI'm going\b

??? >>I'm<< not mad.
ERR \bI'm\b
!!! >>I'm<< not mad.
```

```
??? Now >>I'm<< here to stay
ERR \bI'm\b
SPC Now I'm here to stay
```

Records starting with '???' show the problem; record starting with 'ERR' show the error rule that is responsible; record starting with 'EXC' show an exception rule that revert the error into a valid string; record starting with 'SPC' show a special string that is to be considered valid; record starting with '!!!' show an error that persist.

5 How to install Textchk

Textchk is made essentially of one executable: `textchk`. This file can be placed everywhere you can run it without giving the path; that is: inside a directory listed by the environment variable `PATH`.

It is needed Perl as '`/usr/bin/perl`'. If your system is organized differently, you should modify the first line of this executable:

```
#!/usr/bin/perl
#...
```

After that, you need only a suitable '`./.textchk.rules`' and maybe also '`./.textchk.special`'

5.1 Gettext

The messages that Textchk shows may be translated. To install the already translated PO files, it is necessary to compile them like this:

```
msgfmt -o textchk.mo it.po
```

In this example the file '`it.po`' is compiled and it is generated the file '`textchk.mo`'. This generated file must be copied inside the right directory; in this case, may be '`/usr/share/locale/it/LC_MESSAGES/`'.

If you don't have installed the Perl-gettext module and you don't want to warry about it, you can comment the following instructions:

```
# We *don't* want to use gettext.
#use POSIX;
#use Locale::gettext;
#setlocale (LC_MESSAGES, "");
#textdomain ("textchk");
```

Then you have to introduce a dummy '`gettext()`' function:

```
sub gettext
{
    return $_[0];
}
```

5.2 Dependencies

Textchk depends on other software to transform manual pages, HTML pages and Texinfo sources into normalized text. This is Groff, Lynx and Texinfo. As it is included the use of Gettext, the Perl-gettext module must be installed.

Index