spelling*

Stephan Hennig†

4th December 2012

Abstract

This package aids spell-checking of TEX documents compiled with the LuaTEX engine. It can give visual feedback in PDF output similar to WYSIWYG word processors. The package relies on an external spell-checker application to check spelling of a text file and to output a list of bad spellings. The package should work with most spell-checkers, even dumb, TEX-unaware ones.

Warning! This package is in a very early state. Everything may change!

Contents

1 Introduction

Ther are three main approaches to spell-checking TEX documents:

1. checking spelling in the `.tex` source file,
2. converting a `.tex` file to another format, for which a proved spell-checking solution exists,
3. checking spelling after a `.tex` file has been processed by TEX.

*This document describes the spelling packgae v0.2.

†sh2d@arcor.de

All of these approaches have their strengths and weaknesses. This package follows the third approach, providing some unique features:

- When extracting text from typeset DVI, PS or PDF files, hyphenation has to be switched off during the TEX run to avoid lots of false positives being reported by the spell-checker. That is, one doesn't work on the original document.

  In contrast to that, the spelling package works transparently on the original `.tex` source file. Text is extracted *during* typesetting, after LuaTEX has applied its catcode and macro machinery, but before hyphenation takes place.

- The spelling package can highlight words with known incorrect spelling in PDF output, giving visual feedback similar to WYSIWYG word processors.[1]

2 Usage

The spelling package requires the LuaTEX engine. Currently, only the LATEX format is supported. *The package consists of Lua modules that do the actual work. The LATEX package is effectively a wrapper around the modules' Lua-API. Implementing support for other formats shouldn't be too difficult. Since the author is a LATEX-only user, support has to be contributed by,* e. g.*, you. By the way, the LATEX package needs some polishing, too. Contributions are welcome!*

2.1 Work-flow

The work-flow of the spelling package is as follows:

1. After the package is loaded in the preamble of a `.tex` source file via `\usepackage{spelling}`, a list of bad spellings is read from a file named ⟨*jobname*⟩`.spb`, if that file exists.

2. During the LuaTEX run, text is extracted from pages and all words are checked against the list of bad spellings. Words with a known incorrect spelling are highlighted in PDF output.

[1] Currently, only colouring words is implemented.

3. At the end of the LuaTEX run, in addition to the PDF file, a text file is written, named ⟨*jobname*⟩`.txt`, by default. The text file should contain most of the text of the original document.

4. The text file is then checked by your favourite spell-checker application, *e. g.*, Aspell or Hunspell. The spell-checker should be able to write a list of bad spellings to a file. Otherwise, visual feedback in PDF output won't work. The file should preferably be named ⟨*jobname*⟩`.spb`, but any other file name works just as well.

5. Now, there are two ways to proceed:

   (a) Visually minded people may just compile their document a second time. This time, file ⟨*jobname*⟩`.spb` is read-in again and the words with incorrect spelling found by the spell-checker should now be highlighted in PDF output. Checking the PDF file, the necessary corrections to the `.tex` source file can be applied.

   (b) If you're not interested in visual feedback or if your spell-checker doesn't provide a non-interactive mode, you can as well apply the necessary corrections directly to the `.tex` source file(s), either interactively, during the spell-checker run, or by looking at the final list of bad spellings in an editor (whatever file it was saved to). That way, the benefit of this package is, that spell-checker input has already been processed by LuaTEX, but contains no hyphenated words.

2.2 Word lists

As described above, if a file ⟨*jobname*⟩`.spb` exists, it is loaded by the spelling package. The words found in the file are stored in an internal list of bad spellings and are later used for highlighting spelling mistakes in PDF output. Words from additional files can be appended to the internal list of bad spellings with the `\spellingreadbad` command. Argument is a file name. `\spellingreadbad`
The format of a word list file is one word per line. The same word may occur multiple times. The file must be in the UTF-8 encoding.

In addition to file ⟨*jobname*⟩`.spb`, a second file ⟨*jobname*⟩`.spg` is read, if that file exists. The words found in that file are stored in an internal list of good spellings. Words in the LuaTEX document are only considered spelling mistakes if they occur in the list of bad spellings, but not in the list of good spellings. That is, words in the list of good spellings are never highlighted in PDF output. The list of good spellings can be used to deal with false

positives (words incorrectly reported as bad spellings by the spell-checker). Words from additional files can be appended to the internal list of good spellings with the `\spellingreadgood` command. Argument is a file name. File format is one word per line. The same word may occur multiple times. The file must be in the UTF-8 encoding. `\spellingreadgood`

Note, most spell-checkers provide means to deal with unknown words via additional dictionaries. It is recommended to configure your spell-checker to report as few false positives as possible.

2.3 Highlighting spellling mistakes

Enabling/disabling Highlighting spelling mistakes (words with known incorrect spelling) in PDF output can be toggled on and off with command `\spellinghighlight`. If the argument is `on`, highlighting is enabled. For other arguments, highlighting is disabled. Highlighting is enabled, by default. `\spellinghighlight`

Colour The colour used for highlighting bad spellings can be determined by command `\spellinghighlightcolor`. Argument is a colour statement in the PDF language. As an example, the colour red in the RGB colour space is represented by the statement `1 0 0 rg`. In the CMYK colour space, a reddish colour is represented by `0 1 1 0 k`. Default colour used for highlighting is `1 0 0 rg`, *i. e.*, red in the RGB colour space. *Warning: There's currently no error checking. Make sure, you're applying a valid PDF colour statement!*

2.4 Text output

Text file After loading the spelling package, at the end of the LuaTEX run, a text file is written that contains most of the document text. The text file is no close text rendering of the typeset document, but serves as input for your favourite spell-checker application. It contains the document text in a simple format: paragraphs separated by blank lines. A paragraph is anything that, during typesetting, starts with a `local_par` whatsit node in the node list representing a typeset page of the original document, *e. g.*, paragraphs in running text, footnotes, marginal notes, (in-lined) `\parbox` commands or cells from `p`-like table columns *etc.*

Paragraphs consist of words separated by spaces. A word is the textual representation of a chain of consecutive nodes of type `glyph`, `disc` or `kern`. Boxes are processed transparently. That is, the spelling package (highly

imperfectly) tries to recognise as a single word what in typeset output looks like a single word. As an example, the LaTeX code

```
foo\mbox{'s bar}s
```

which is typeset as

foo's bars

is considered two words *foo's* and *bars*, instead of the four words *foo*, *'s*, *bar* and *s*.[2]

Enabling/disabling Text output can be toggled on and off with command `\spellingoutput`. If the argument is `on`, text output is enabled. For other arguments, text output is disabled. Text output is enabled, by default. `\spellingoutput`

File name Text output file name can be configured via command `\spellingoutputname`. Argument is the new file name. Default text output file name is ⟨*jobname*⟩`.txt`. `\spellingoutputname`

Line length In text output, paragraphs can either be broken into lines of a fixed length or paragraph can be put on a single line. The behaviour can be controlled via command `\spellingoutputlinelength`. Argument is a number. If the number is smaller than 1, paragraphs are put on a single line. For larger arguments, the number specifies the maximum line length. Note, lines are broken at spaces only. Words longer than maximum line length are put on a single line exceeding maximum line length. Default line length is 72. `\spellingoutputlinele`

Line ending convention The end-of-line (EOL) character in text output can be configured via command `\spellingoutputeol`. Argument is an arbitrary sequence of characters in the UTF-8 encoding. `\spellingoutputeol`

Well, things are a bit more complicated, because in LuaTeX, as in the original TeX, some characters are treated special. To keep LuaTeX from messing with our EOL characters in the input, we need to set their category codes accordingly. As an example, to set EOL character to follow DOS line ending convention (carriage return followed by line feed, TeX notation `^^M^^J`), the following code can be used:

[2] This document has been compiled with a custom list of bad spellings, which is why the word *foo's* should be highlighted.

```
\begingroup
\catcode`\^^J=12% make line feed and carriage return
\catcode`\^^M=12% of category Other
\spellingoutputeol{^^M^^J}
\endgroup
```

For the UNIX line ending convention (a single line feed), just leave out `^^M` in the argument to `\spellingoutputeol`.

Default line ending convention depends on the operating system determined by LuaTEX. If `os.type` is either `windows` or `msdos`, DOS line ending convention is used. Otherwise UNIX line ending convention is used.

2.5 Text extraction

Enabling/disabling Text extraction can be enabled and disabled in the document via command `\spellingextract`. If the argument is `on`, text extraction is enabled. For other arguments, text extraction is disabled. The command should be used in vertical mode, *i. e.*, outside paragraphs. If text extraction is disabled in the document preamble, an empty text file is written at the end of the LuaTEX run. Text extraction is enabled, by default. `\spellingextract`

Note, text extraction and visual feedback are orthogonal features. That is, if text extraction is disabled for part of a document, *e. g.*, a long table, words with a known incorrect spelling are still highlighted in that part.

2.6 Code point mapping

As explained in subsection 2.4, the text file written at the end of the LuaTEX run is in the UTF-8 encoding. Unicode contains a wealth of code points with a special meaning, such as ligatures, alternative letters, symbols *etc.* Unfortunately, not all spell-checker applications are smart enough to correctly interpret all Unicode code points that may occur in a document. For that reason, a code point mapping feature has been implemented that allows for mapping certain Unicode code points that may appear in a node list to arbitrary strings in text output. A typical example is to map ligatures to the characters corresponding to their constituting letters. The default mappings applied can be found in Table 1.

Additional mappings can be declared by command `\spellingmapping`. This command takes two arguments, a number that refers to the Unicode code point, and a sequence of arbitrary characters that is the mapping target. The code point number must be in a format that can be parsed by Lua. The characters must be in the UTF-8 encoding. `\spellingmapping`

Unicode name	code point	target characters
LATIN CAPITAL LIGATURE IJ	0x0132	IJ
LATIN SMALL LIGATURE IJ	0x0133	ij
LATIN CAPITAL LIGATURE OE	0x0152	OE
LATIN SMALL LIGATURE OE	0x0153	oe
LATIN SMALL LETTER LONG S	0x017f	s
LATIN CAPITAL LETTER SHARP S	0x1e9e	SS
LATIN SMALL LIGATURE FF	0xfb00	ff
LATIN SMALL LIGATURE FI	0xfb01	fi
LATIN SMALL LIGATURE FL	0xfb02	fl
LATIN SMALL LIGATURE FFI	0xfb03	ffi
LATIN SMALL LIGATURE FFL	0xfb04	ffl
LATIN SMALL LIGATURE LONG S T	0xfb05	st
LATIN SMALL LIGATURE ST	0xfb06	st

Table 1: Default code point mappings.

New mappings only have effect on the following document text. The command should therefore be used in the document preamble. As an example, the character `A` can be mapped to `Z` and *vice versa* with the following code:

```
\spellingmapping{65}{Z}% A => Z
\spellingmapping{90}{A}% Z => A
```

Another command `\spellingclearallmappings` can be used to remove all existing code point mappings. `\spellingclearallmapp`

2.7 Tables

How do tables fit into the simple text file format that has only paragraphs and blank lines as described in subsection 2.4? What is a paragraph with regards to tables? A whole table? A row? A single cell?

By default, only text from cells in `p`(aragraph)-like columns is put on their own paragraph, because the corresponding node list branches contain a `local_par` whatsit node (*cf.* subsection 2.4). The behaviour can be changed with the `\spellingtablepar` command. This command takes as argument a number. If the argument is 0, the behaviour is described as above. If the argument is 1, a blank line is inserted before and after every table row (but at most once between table rows). If the argument is 2, a blank line `\spellingtablepar`

is inserted before and after every table cell. By default, no blank lines are inserted.

3 Bugs

Note, this package is in a very early state. Expect bugs! Package development is hosted at **GitHub**. The full list of known bugs and feature requests can be found in the **issue tracker**. New bugs should be reported there.

The most user-visible issues are listed below:

- There's no support for the Plain TEX or ConTEXt formats other than the API of the package's Lua modules, yet (issue 1).
- Macros provided by the LATEX package have very long names. A *key-value* package option interface would be much more user-friendly (issue 2).
- There are a couple of issues with text extraction and highlighting incorrect spellings:
  - Text in head and foot lines is neither extracted nor highlighted (issue 7).
  - Punctuation characters are currently not stripped from words. If the list of bad spellings contains an entry, say, *incorect*, and the text contains that spelling directly followed by a comma, a quotation mark *etc.*, the incorect spelling is not highlighted (issue 8).
  - The first word starting right after an `hlist`, *e. g.*, the first word within an `\mbox`, is never highlighted. It is extracted and written to the text file, though (issue 6).
  - Bad spellings that are hyphenated at a page break are not highlighted (issue 10).

Any contributions are warmly welcome!

Happy TEXing!