

# Beyond spellchecking - What else can we check automatically

By Tibs / Tony Ibbs (they / he)

Slides and accompanying material at  
<https://github.com/tibs/beyond-spellchecking>

*tony.ibbs@aiven.io / @much\_of\_a*

# Introduction

Writing documentation is hard, and spotting errors in that documentation is harder. What can we do to help?

- Automated checking: "Linting"
- A review of checks we might make
- Adopt existing checks or grow your own
- Some tools
- Plumbing checks into docs-as-code
- What we do at Aiven

*tony.ibbs@aiven.io / @much\_of\_a*

# "linting"

`lint` was a program written in 1978 to find common errors and stylistic problems in C code

`lint` = bits of fluff on fabric, hence *linting*, to remove them

Simple checks, that can be fast, and give good results

But remember, text is not as restricted as code

# Types of check

Not an exhaustive list, but with example messages

*tony.ibbs@aiven.io / @much\_of\_a*

# Spelling

`'arglebargle' does not seem to be a word`

Ignore numbers, some punctuation...

Can only report mistakes

*tony.ibbs@aiven.io / @much\_of\_a*

# 🤔 But I really did mean arglebargle!

For this, we shall invent the term 'arglebargle'.

What to do?

- Put up with it
- Mark it up differently (e.g., as "literal" text)
- Configure in the text (`.. lint: off / .. lint: on`)
- Configure to ignore `arglebargle`
- Configure to ignore `arglebargle` *in this file / location*

*tony.ibbs@aiven.io / @much\_of\_a*

## Against auto-correction

Written text is complicated, and linting will find false positives

Auto-correction can lead to unexpected results

The final decision should be with a human

## Don't say that

Consider not using 'it is obvious that'





## Use *this* instead of *that* - errors

Use 'and' instead of 'adn'

Use 'supersede' instead of 'supercede'

Use 'Aiven for Redis' instead of 'Aiven Redis'

## ✖ Use *this* instead of *that* - warnings

Consider using 'flink' instead of 'flick'

Consider using 'for instance' instead of 'e.g.'

# Errors versus warnings

An error must be fixed, the document is wrong

A warning is just a warning - a "suggestion"

What do you do after you get a warning?

🤔 Create tests you need, retire them when not

If the person who mistypes `adn` leaves the team

You probably don't still need the check for `"adn"` -> `"and"`

*tony.ibbs@aiven.io / @much\_of\_a*

## ✖ One or the other, not both

Inconsistent spelling of 'center' and 'centre'

✖ If *this* is present, then we need *that*

WHO has no definition

At least one use of 'PostgreSQL' must be marked as ®

*tony.ibbs@aiven.io / @much\_of\_a*

# 🤔 word versus token versus ...

`word` - like in a dictionary

`token` - like in a parser, more general

`expression` - like a regular expression, a pattern to match

## 🤔 scope

"Scope" - some part of a document

- Thing must be used with ® in the first *title* to use the name
- Thing must be used with ® in the first *non-title* to use the name
- Thing must be used with ® the very first time it occurs

*tony.ibbs@aiven.io / @much\_of\_a*



# Capitalisation

'Badly Capitalised Heading' should be in sentence case

But consider carefully:

iPhone prices

The importance of NASA

Remembering Terry Jones

*tony.ibbs@aiven.io / @much\_of\_a*

# Checking beyond the text itself

Some tests aren't just looking at the text of the document

*tony.ibbs@aiven.io / @much\_of\_a*

## ✖ Looking at the raw text

### Checking reStructuredText:

```
One backtick without a role becomes italics
```

```
Use reStructuredText link format, not Markdown
```

### Checking Markdown:

```
Two backticks is redundant - did you mean just one?
```



# Checking for absence

For instance, that all images have `alt` text

```
Image is missing alt text
```

Not the same as "is zero length" - we want *structural element* occurs zero times

In HTML, `alt=` in ``

In reStructuredText, `:alt:` inside image directive

## Arbitrary metrics

Try to keep the Flesch-Kincaid grade level (12) below 8

This is calculated as something like

$$(0.39 * (\text{words} / \text{sentences})) + \\ (11.8 * (\text{syllables} / \text{words})) - 15.59$$

*tony.ibbs@aiven.io / @much\_of\_a*

# Sentence analysis

## NLP (Natural Language Processing)

Did you mean "cars are" instead of "car's are"

Don't use "like" as an interjection

## ✖ Just let me code

Writing a plugin with access to knowledge of the document structure

*tony.ibbs@aiven.io / @much\_of\_a*

# What to use for the task

*tony.ibbs@aiven.io / @much\_of\_a*



# Pre-built or hand-designed

Adopt an existing package - Microsoft or Google styles

Do that and add customisations

Start from scratch and specify everything yourself

...but if you do your own checks, consider contributing back to the community

*tony.ibbs@aiven.io / @much\_of\_a*

# Available tools

Just a brief overview...

- Vale
- LTeX and LanguageTool
- alex
- proselint
- RedPen
- textlint

*tony.ibbs@aiven.io / @much\_of\_a*

# Vale

Vale supports checking in Markdown, HTML, reStructuredText, AsciiDoc, DITA, XML, Org and code (comments / docstrings).

Rules ("styles") are specified via YAML files that build on existing concepts, or (less often) via code in a Go-like language

Various pre-packaged rulesets are available

*tony.ibbs@aiven.io / @much\_of\_a*

# LT<sub>E</sub>X and LanguageTool

LT<sub>E</sub>X provides offline grammar checking of various markup languages using LanguageTool

BibT<sub>E</sub>X, ConT<sub>E</sub>Xt, LaT<sub>E</sub>X, Markdown, Org, reStructuredText, R Sweave, and XHTML

English, French, German, Dutch, Chinese, Russian, etc.

New rules for LanguageTool are stored as XML files

# alex

**alex** is designed to "Catch insensitive, inconsiderate writing" in Markdown documents, and offer alternatives

*tony.ibbs@aiven.io / @much\_of\_a*

# proselint

proselint runs checks on Markdown files

It comes with its own set of checks built in

New checks are written as plugins using Python

*tony.ibbs@aiven.io / @much\_of\_a*

# RedPen

RedPen validates texts in Markdown, Textile, AsciiDoc, reStructuredText and LaTeX

It supports multiple languages, including English, German, Japanese and Chinese

There is a catalogue of existing validators to choose from, and custom validators can be written as plugins in Java or JavaScript

*tony.ibbs@aiven.io / @much\_of\_a*

# textlint

**textlint** supports Markdown and plain text by default, with plugins for HTML, reStructuredText, AsciiDoc, Re:VIEW and Org-mode

Starts with no rules installed, use `npm` to install rules by name.

New rules are written as plugins using JavaScript



# Plumbing checks into docs-as-code

*tony.ibbs@aiven.io / @much\_of\_a*

# Local checks

In the editor - display messages as you're typing, or on saving

At the command line - run a command to make the checks

*tony.ibbs@aiven.io / @much\_of\_a*

# Checks before commit

Don't allow `commit` if there are errors

*This may be a bit extreme?*

# Checks before review

Run checks when change are pushed for review

The reviewers can see the results

Forbid merging if there are errors?

*Seems more reasonable*

On GitHub, use workflows for this

# Checks before deployment

Don't deploy if there are errors

*Probably a good idea* - **if** the previous stages mean this essentially never happens

# Use in CI (continuous integration)

Run the checks automatically when a review is requested (GitHub: PR) or before deploying the documentation

No errors before deployment...

# What we do at Aiven

We lint Aiven's developer documentation

<https://docs.aiven.io/> and <https://github.com/aiven/devportal>

*tony.ibbs@aiven.io / @much\_of\_a*

# We use Vale

- It's a small program, it's fast, it's portable, it's very configurable
- Development is ongoing, the code is readable, the author fixes bugs quickly
- It's well known in the WtD community

But...

- It's a relatively small project
- We did (do) need to configure it

*tony.ibbs@aiven.io / @much\_of\_a*



# The checks we use

- spelling - Spell checking - the default US-en dictionary, plus our own
- capitalization - Capitalisation in headings
- substitution - Use *this* instead of *that*
- conditional - If *this* then *that*, for ® checking

# At the command line

```
make spell
```

# In CI (continuous integration)

We use [vale-action](#), the official GitHub action for Vale

We run checks:

- For a PR (pull request)
- When pushing to `main` (in theory...)



## What have we learnt?

- We can check things beyond spelling
- Relatively simple techniques can be useful
- But don't check for the sake of it
- There is a good choice of tools available
- You don't have to build it yourself
- You can check as part of your docs-as-code toolchain

# Fin

Come join us on [Write the Docs slack](#) channel  
[#testthedocs](#)

Slides and accompanying material at  
<https://github.com/tibs/beyond-spellchecking>

Written in [reStructuredText](#), converted to PDF using  
[rst2pdf](#)



This slideshow is released under a [Creative Commons Attribution-ShareAlike 4.0 International License](#)



*tony.ibbs@aiven.io / @much\_of\_a*