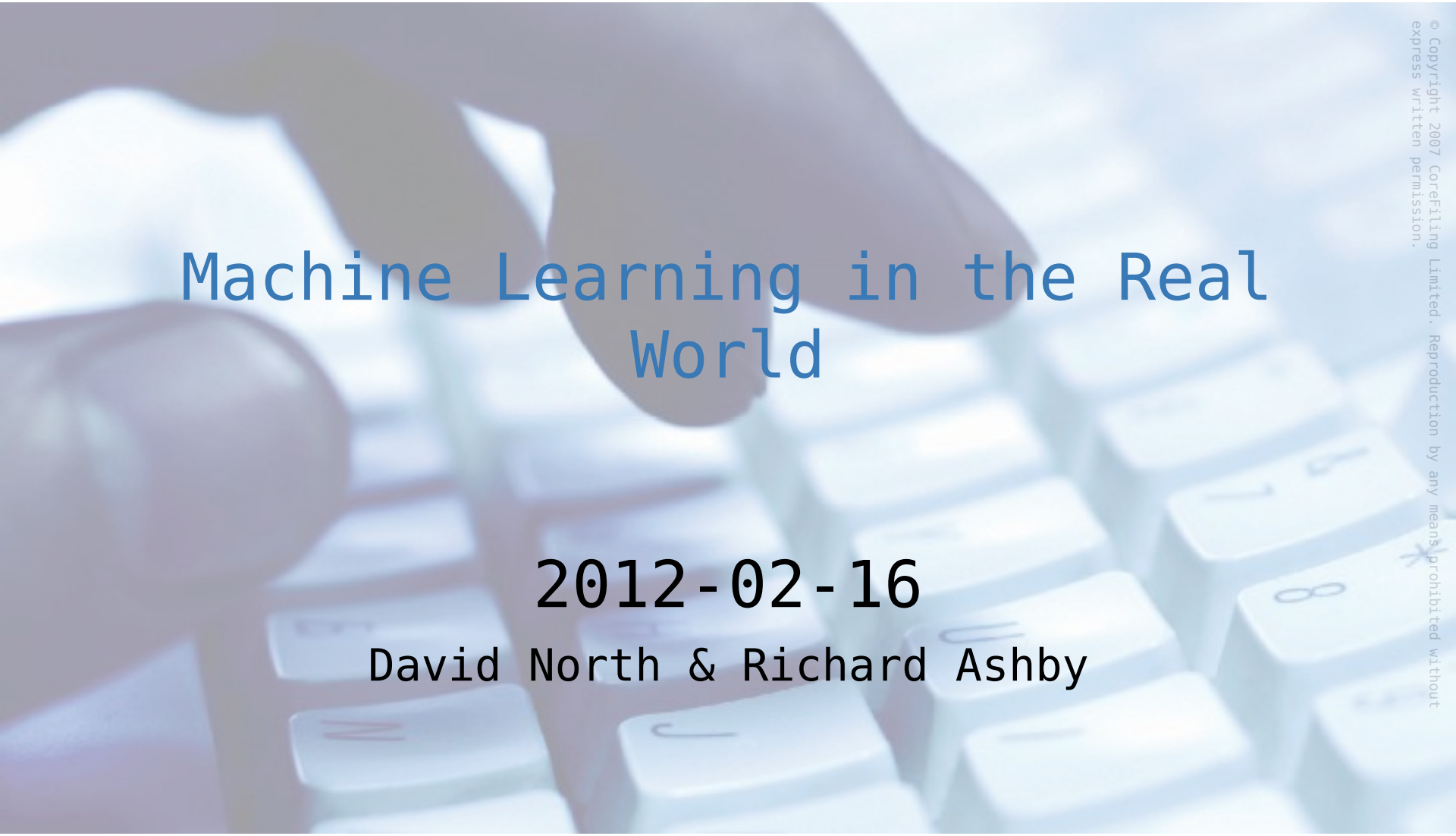


Machine Learning in the Real World

2012-02-16

David North & Richard Ashby

CoreFiling

The company

- CoreFiling is a software company with main office in Oxford
- 50 people, 20 developers
- Financial/business reporting software
- Mostly Java, some Scala, Python, .NET, always willing to use the best tool for the job

Us

- The amusingly(?) named 'Team R&D', the smallest dev team in the company
- Have worked together on various projects over the last 3 years
- David – CS at Oxford (06-09), intern at CFL, liked it so much he came back

Richard

- Classics at Oxford 1997-2002
- CS at Sydney, 2006-07
- At CoreFiling since 2007

Why we're here

Because we want to recruit people!

Because what we're talking about is an interesting cross-over between industry and academia

The Problem

From April 2012, all UK corporation tax returns must be submitted to HMRC in XML.

- Previously, submissions have been on paper or as PDFs
- Current process and tools are geared towards producing accounts as MS Word documents

The long-term solution

Accounting software vendors need to fix their software to allow export of annual returns in the XML format required by HMRC

Meanwhile...

- This will take years to implement and trickle down to users via upgrades
- Lots of small firms do their accounts in Excel (or even on paper)
- A stop-gap solution is called for (and is a massive opportunity for us)

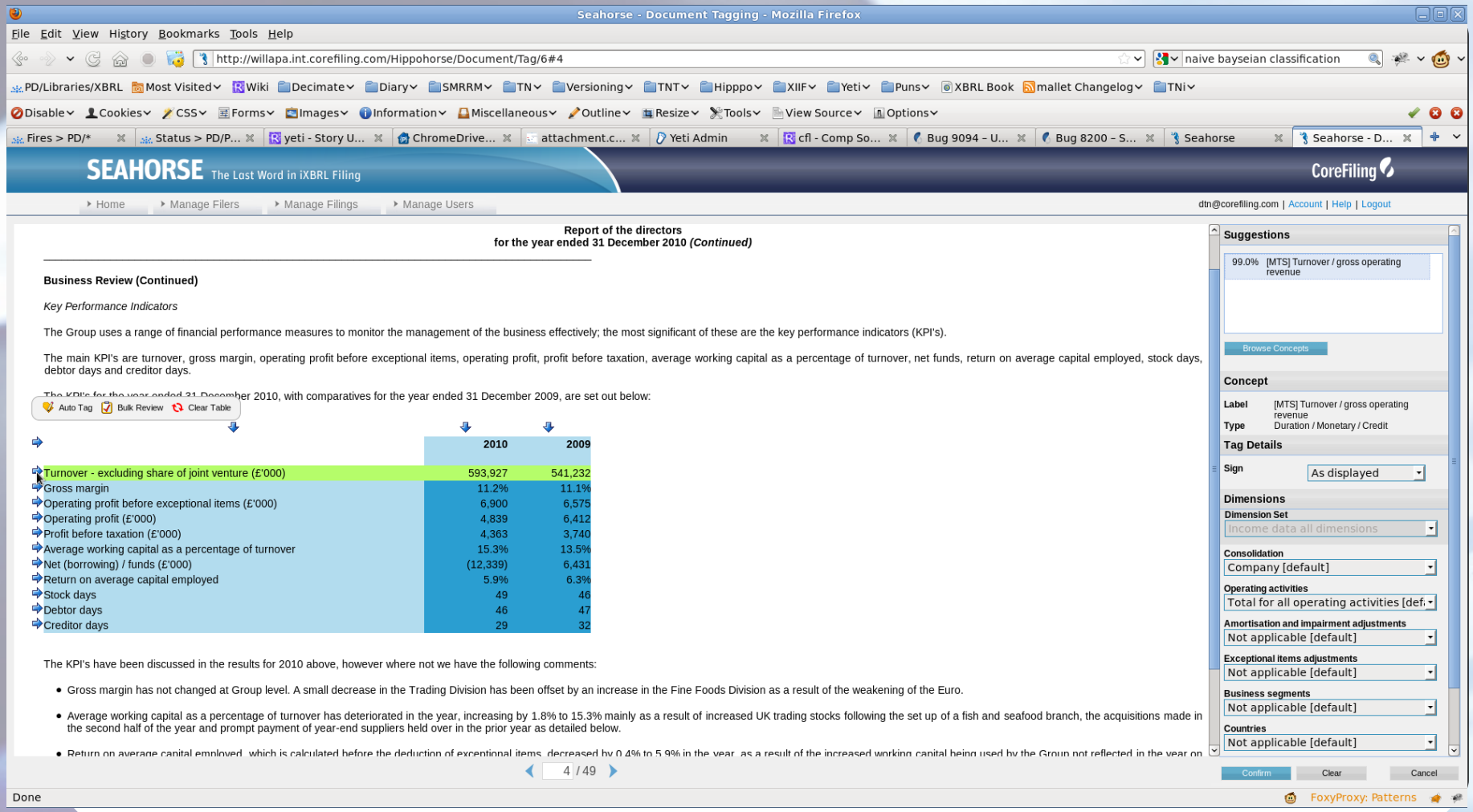
Tagging

- Take the existing Word document
- Have the user (an accountant) go through and 'tag' the items HMRC expects to be reported
- Use the tags to export the necessary XML

Seahorse

A web app which allows users to upload and tag their Word (or Excel) accounts, then export the XML.

We host this and sell it as SASS (Software As A Service)



CoreFiling

Sub-problem

- Tagging is **very** tedious
- Could be several hundred table rows and text items in one filing
- One person could be tasked with preparing dozens of these
- Not feasible to do by hand

Machine Learning

Filings are of broadly similar shape to each other

(especially those all prepared by the same accountancy firm)

Can we use machine learning to identify where the tags need to go?

For legal/regulatory reasons, a human must review anything suggested automatically, which allows us to use *supervised* machine learning methods – those where the right answer is externally confirmed and can be used to improve the suggestions in future

Scoping the problem

Can potentially use machine learning for:

* Identifying tables in the document
* Identifying facts in chunks of text
* Picking the tag on a given chunk of text

Decided to only attempt the last of these – parsing unstructured data is very difficult to get right, and most of what we want to tag is in easily identifiable MS Word tables (or Excel)

Formalize the problem

Given a table, and the list of 'concepts' (possible HMRC-defined tags), suggest tags for each taggable row.

Need to define 'taggable'

This is a **classififcation** problem, well understood in principle (flowers into species etc).

A table to be tagged

	2010	2009
Turnover - excluding share of joint venture (£'000)	593,927	541,232
Gross margin	11.2%	11.1%
Operating profit before exceptional items (£'000)	6,900	6,575
Operating profit (£'000)	4,839	6,412
Profit before taxation (£'000)	4,363	3,740
Average working capital as a percentage of turnover	15.3%	13.5%
Net (borrowing) / funds (£'000)	(12,339)	6,431
Return on average capital employed	5.9%	6.3%
Stock days	49	46
Debtor days	46	47
Creditor days	29	32

“Good programmers don't write what they can steal”

(and good companies don't pay for tools available as FOSS)

Are there any libraries for solving classification problems available as free/open source software?

FOSS Machine Learning Tools

All developed in academia as one might expect...

Weka – as used in various CS machine learning practicals. Works very well, has one serious problem for us...

Hadoop – top-level Apache machine-learning project, lots of commercial backing and use, but v. complicated, big and heavy (more for Google and Yahoo than us)

Mallet – the happy medium.

Classification methods

Mallet has a common API allowing us to swap in and out different methods easily (just change a couple of lines of code)

Naive Bayesian Methods

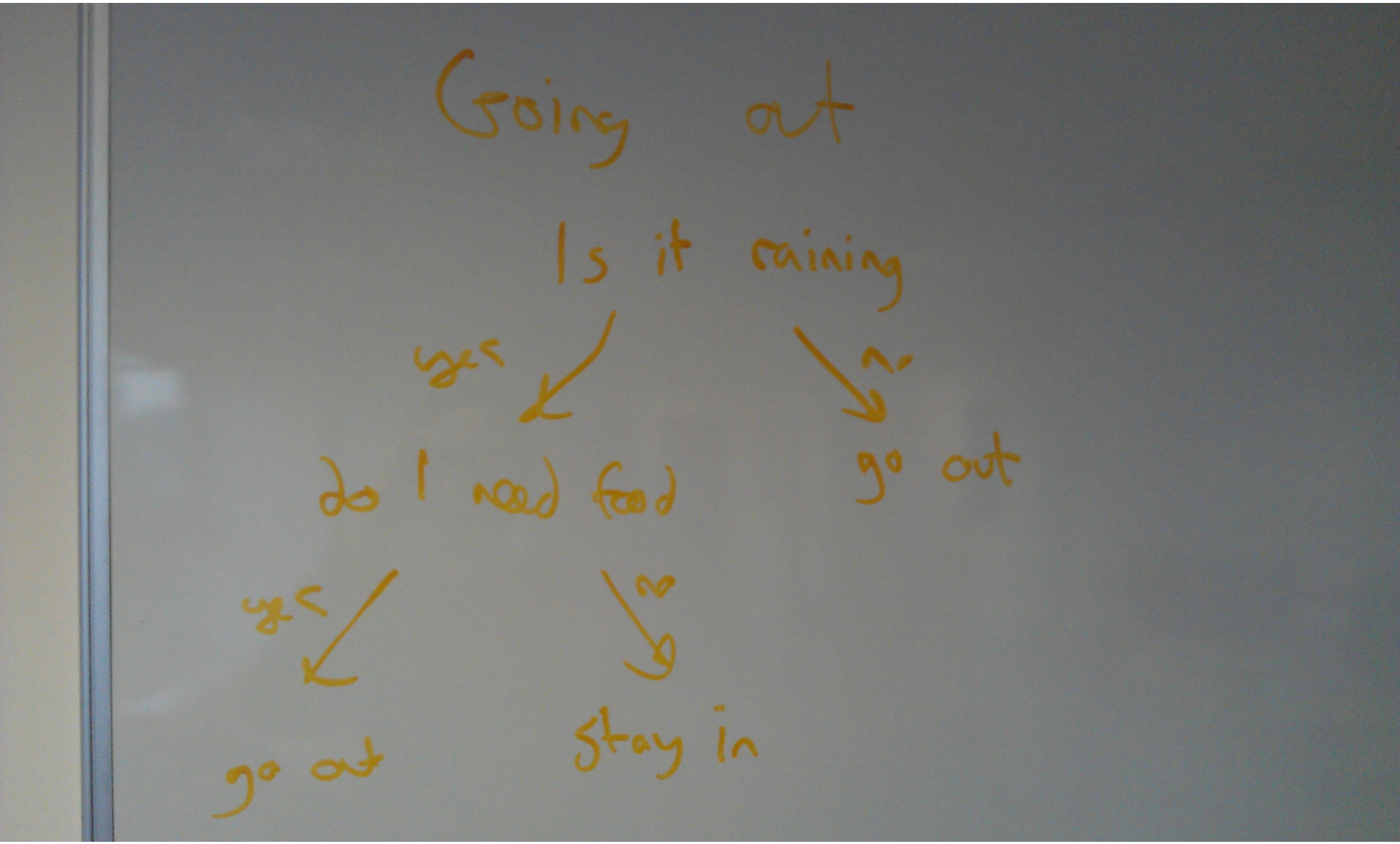
As used by your spam filter

Assumes 'feature independence'

eg. Trying to decide if fruit is an apple, assume shape, size, taste are independent predictors (not related), take the product of probabilities from each of these

- Works great on spam and other problems, especially yes/no questions
- Computationally cheap
- However, intuitively, for our problem, the assumption is rubbish
- This is borne out by poor test results

Decision Trees



Maximum Entropy Modelling

(with apologies for hand-waving)

“Assume only as much as our available information tells us”

Classification of a given row is a discrete probability distribution over n mutually exclusive propositions (tags)

In the absence of any information, assume all known tags are equally likely

As we gather training data, we want our prob. distribution to 'agree' with it – various means of extracting constraints from the data

Having extracted our constraints, brute-force the set of possible

prediction functions satisfying them until we find the one which gives the maximum entropy on the available data.

'Brute force' can actually be refined to a spatial search in n dimensions. Quite expensive (time/memory) to do, but model cheap to use once built.

How do we know it's working?

No maths/formal methods

Test-driven development

Quantify the desired behaviour (using our tame accountants)

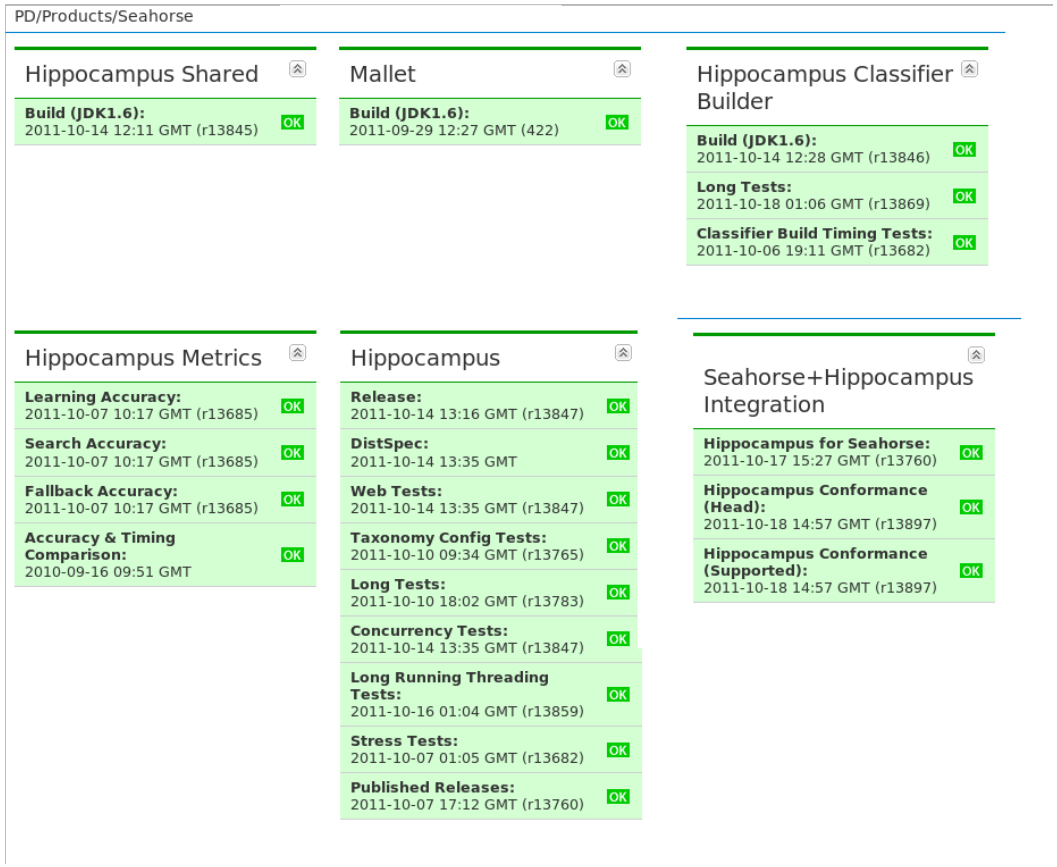
Write automated tests capturing this behaviour

Initially, they'll fail (we haven't written any code yet)

Refine the implementation until the tests pass and the customer is satisfied wrt the original spec

Guard against regression by running the tests automatically (not just nightly, all the time – continuous integration)

Yay, lots of tests all passing



CoreFiling

Not just pass/fail

- In the case of this problem, measure accuracy as a percentage and plot a graph.
- Cross-fold testing: take known tagged documents, leave one out, train with others and test with this one
- Take the mean result

Chosing our metrics

The obvious: how often is the exact right answer selected?

How often is the right answer in the top ten?

How often are we under-confident in the right answer or over-confident in the wrong one?

As it turns out...

- Right answer is in top 10 suggestions 78% of the time (mean x-fold across available data)
- Right answer is top suggestion 63% of the time
- This took two people just over a year (~ 2,000 man hours) to achieve