Agile Data Science

Agile Alliance Technology Conference
April 2017

Terran Melconian
terran@airnetsim.com
twitter @terranmelconian

Same as Engineering?

- Both involve technical work, writing code, and interacting with businesspeople!

- Effort usually not foreseeable
- Achievable results usually not foreseeable
- Most work is exploration which is discarded

“Data science”?

- Warehousing
- Business Intelligence & Reporting
- Offline Modelling
- Algorithm Development
- Research

- This presentation will motivate guidelines by following our hapless data protagonist through numerous bad decisions.

Examples for Presentation

- We work at a company which sells lots of distinct products to consumers or small business.
  - Self-service purchase
  - Search functionality prominent on site
  - Email signup
  - Repeat customers

- We thought up two possible projects:
  - Spelling Correction on Searches
  - Personalized Recommendations

Can we ask for priorities?

- Let’s go ask somebody in the business which project to do.
- They might need some information in order to decide…
  - “How long will it take?” “Sorry, don’t know.”
  - “What benefits will it produce?” “Uh….”
- They say: Do whatever will best maximize sales.
- Lesson: Data science projects are fungible.

Failure by Success

- I built a state of the art spelling correction system…
- It supports English, French, Italian, German, and Spanish…
- It includes heuristics for keyboard distance…
- It outperformed the other spelling correction systems in an online competition…
- It took four months to build and integrate…
- It turns out that 0.3% of our searches involve misspellings and the revenue is negligible…
- Lesson: Know your best case results and benefits.

Estimating Spelling Correction Benefits

- Do we *have* a log of what the user typed?
- How do we tell when it's a mistake?
- How many searches have spelling mistakes?
- What's the downstream consequence of a misspelling on sales?
- How many of the mistakes can we correct?
- Result: estimate that spelling correction will make $$$$/week in new sales.

Implementing Spellcorrect

- OK, I verified my benefits, so NOW I can take months and build my super-duper spelling correction system, right?
- Oops! It turns out that 95% of the people who made spelling mistakes fixed them and did a new search themselves anyway.
- or… Oops! It turns out that 90% of our spelling corrections were on mobile and mobile traffic converts much less than the average we used.
- Lesson: You never fully understand the dynamics.

Implementing Intelligently

- What's the easiest implementation we can possibly do which will get a measurable fraction of the result?
  - Time to do some stats on what's measurable…
  - Lookup table for most common mistakes?
  - Single character correction in our search engine?
- **Deploy the minimum viable feature and measure it!**

Prioritization

- OK, my test worked, so NOW I can finally build my nice system, right?
- Wrong! Can you get 80% of the benefit in 20% of the time with the lookup table or another simple solution?
- We haven't even started looking at recommendations, after all!
- Lesson: You have more good ideas than time.

Quick Test for Recs

- How can we evaluate the benefits and test as quickly as possible?
  - Unlike spelling, there's no clear post-hoc assessment.
- Deploy a quick test?
  - For popular searches, where there's plenty of data, show what people ultimately bought
  - Engineering release cycle - could be weeks
- How about those email subscribers?

More Recommender Tests

- Verified on email, so now do the engineering.
- Good results on the first test, so now we want to expand who sees recommendations… go through the engineering cycle all over again?
- Instead of one fixed implementation, deploy a system for reading recommendation from a file or DB which can be updated with a faster process.
- Lesson: Design for fast iteration

A/B Tests

- We got great results when we rolled out our recommender, but it turns out we also introduced a new product line the same week, so how do we attribute the gains? Both teams claim credit…
- Longitudinal tests are difficult and ambiguous to interpret.
- We need to show the recommender to some people and not to others, who are otherwise identical, to get clean results.
- Lesson: Need A/B tests, AKA randomized trials.

Team Structure

- Who's going to do the engineering to make the A/B tests or a recommender?
  - Can you do it yourselves? May you do it yourselves?
- Do you have engineers allocated?
  - Now you're the product owners
- None of the above?
- Lesson: Know where your engineering is

Implementation Quality

- We wrote this to throw away, but it turns out it worked. Now what?
- Write it again, better
  - Design it for fast iteration
- **Reliability and robustness? Think 9s**
  - How often does it fail?
  - What does that failure cost?
  - How does reducing that failure cost compare against the next best project?

Managing the Backlog

- Recurring areas: do we work on recommendations or spelling?
- Now it’s time to record your estimates and compare against the actual result to calibrate.
- A likely productivity curve:
  - Increases at beginning with increasing domain knowledge and tooling
  - Decreases again after best and easiest ideas have been implemented
- Analogies: A*, multi-armed bandit

Opening New Areas

- Would be convenient to start on new areas when returns diminish on existing projects
  - Context switching has costs
- Collecting new data has a long lead time
  - Need to start collecting well in advance
- Adding new features will have business and product interactions
- Manage a pool of exploitable domains
  - Kanban-style

Other Business Goals

- Special projects
  - “We’re now also selling things for partners. Figure out where to show them to maximize profit while keeping our partners happy.”
- Strategic goal changes
  - “This quarter, we care about time on site and number of page views in addition to profit.”

Software Testing & Maintenance

- Data validation
  - Do we have the right amount?
  - Are certain cases for which we KNOW we should have data covered?
  - Does the data pass some sanity tests (smoke tests)?
- Holdbacks
  - Keep 1% to 10% of users/arrivals/etc permanently out of the feature.
  - When rolled out, it produced X% profit lift. Does it still produce that compared to the baseline?

Summary of Lessons (1)

- The team should get the desired result(s) from the business, and then choose the most effective path.
- Estimate your best case outcome at each stage, using what you have measured and learned.
- Proceed incrementally, doing the minimum possible work to observe the next set of unknowns.

Summary of Lessons (2)

- After each step, be willing to say “good enough” or “not worth it” and prioritize a different project.
- Build for fast iteration.
- Demand an A/B test (controlled trial) framework.
- Engineering work is critical.

Questions?

rotten fruit?

terran@airnetsim.com

twitter @terranmelconian

If you're reading an archive of this presentation,
I'm happy answer questions by email.