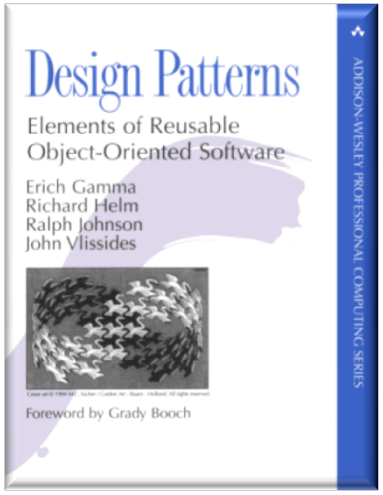
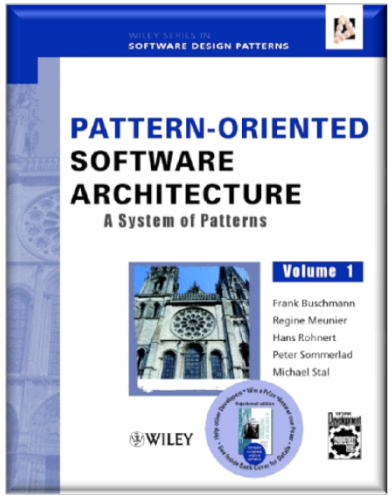
Patterns (and Anti-Patterns) for Developing Machine Learning Systems

Gordon Rios
(gparker@gmail.com)
Zvents, Inc.
Hypertable.org

Patterns and Anti-Patterns

- Strategic, tactical, and operational
- Anti-Patterns – seems obvious but is actually questionable or a "bad" idea
- References: Design Patterns (Gamma, et al.) and Pattern Oriented Software Architecture (Buschmann, et al.)

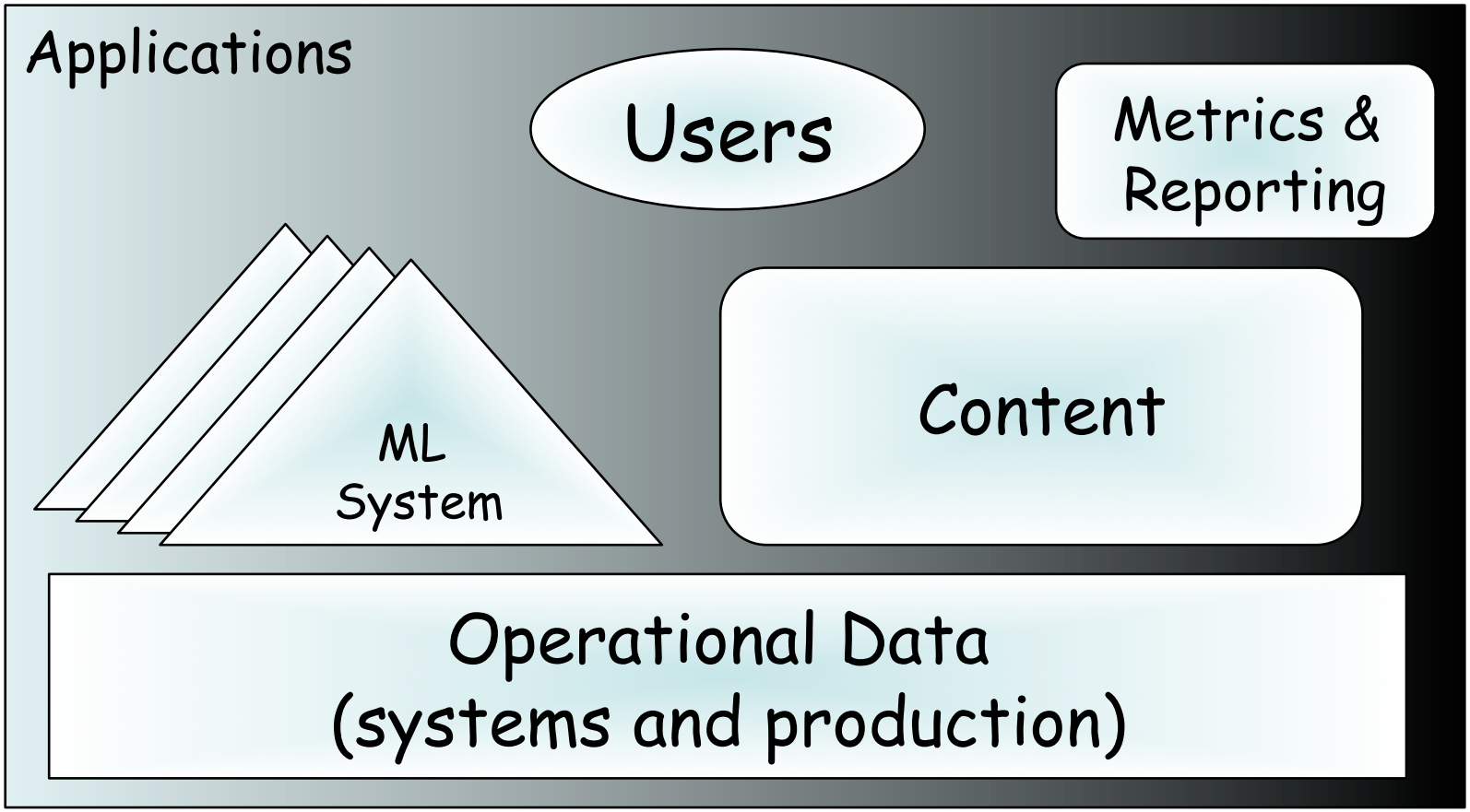




Trapped in the Maze

- ML projects are complex and disruptive
- Ownership distributed across organization or missing completely
- Political factors can create a maze of dead ends and hidden pitfalls
- Familiarity with ML is sparse at best

A Simple Context

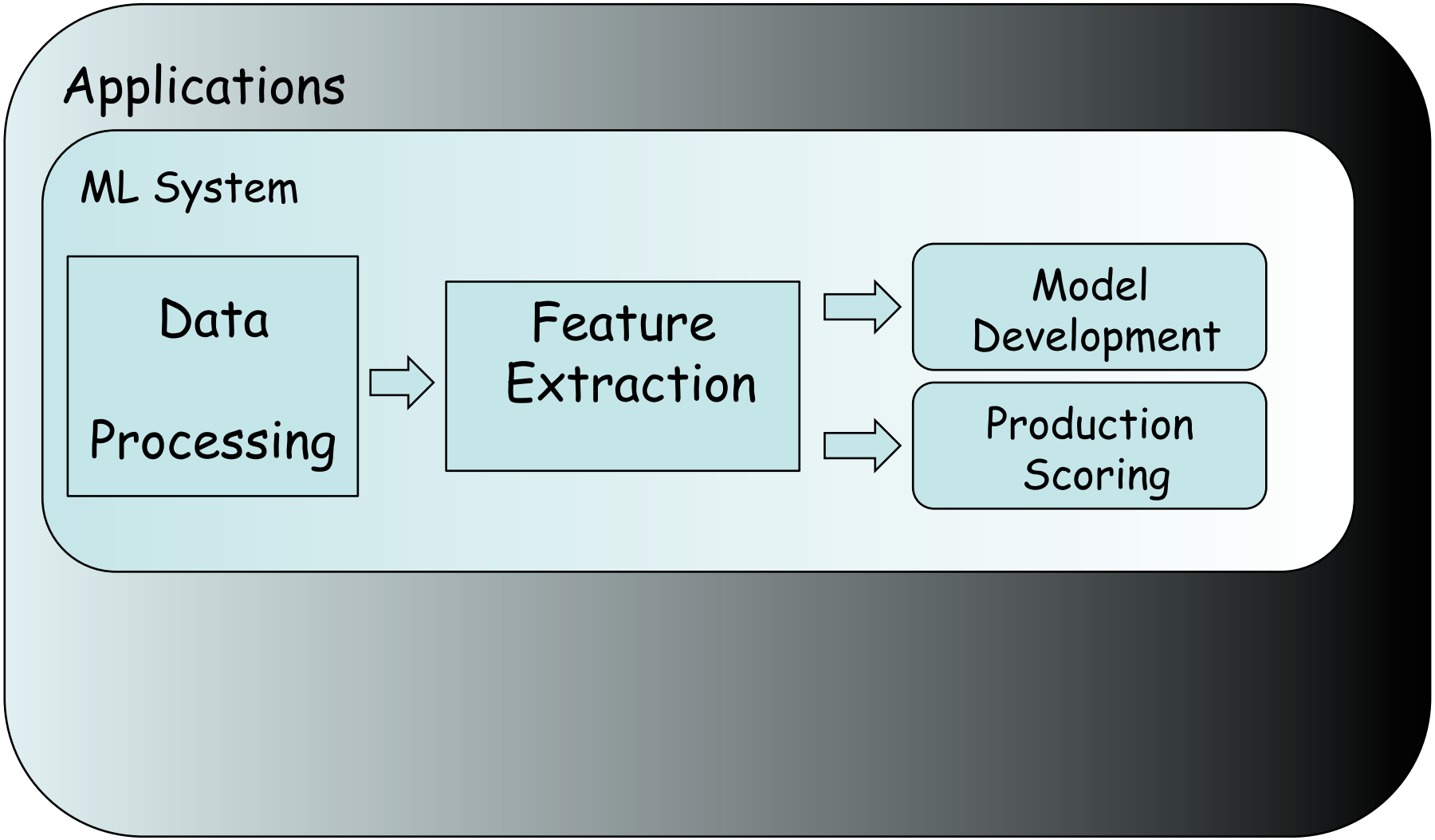


"Stuck" on top of the Pyramid

Level of effort:

1. Data processing systems at the base
2. Feature engineering in the middle
3. Models stuck at the top and dependent on all the rest …

Basic Components the ML System

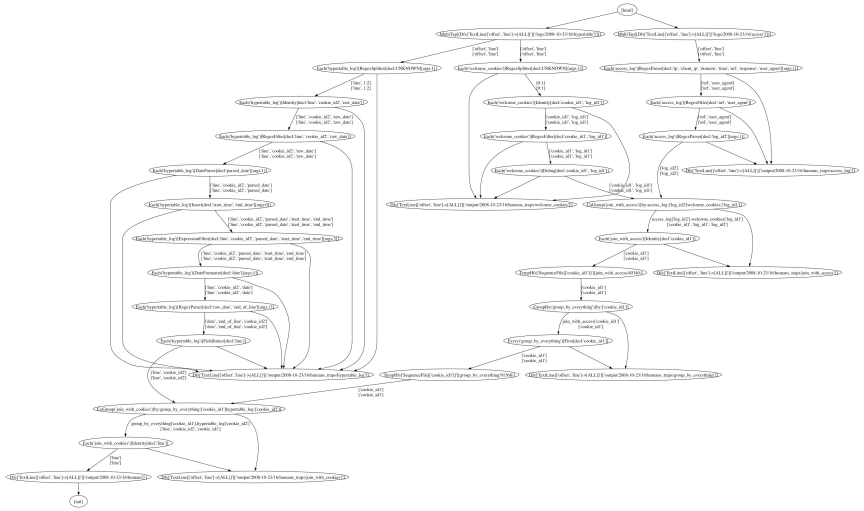


Thin Line (of Functionality)

- Navigate safely through the negative metaphors
- Encounter potential issues early enough in the process to manage or solve
- Keep each piece of work manageable and explainable
- Caution: if your thin ML system is "good enough" organization may lose interest in more advanced solution (80/20)

Workflow

- Data and operations are messy – mix of relational database, logs, map-reduce, distributed databases, etc.
- Think and plan in terms of workflows and be aware that job scheduling is hidden complexity for map-reduce
- Use tools such as cascading http://www.cascading.org)
- Related: Pipeline



Legacy

- An older model or early approach needs to be replaced but has entrenched support
- Use as an input to new approach (presumably based on ML)
- Can be technically challenging but frequently can be converted to an input in conjunction with Pipeline
- Related: Chop Shop, Tiers, Shadow
- Advanced: Champion/Challenger

Shadow

- Legacy system is an input to critical processes and operations
- Develop new system and run in parallel to test output or regularly audit
- Can be used as sort of Champion /Challenger-lite in conjunction with Internal Feedback
- Also apply to upgrades to input pipeline components

Chop Shop

- Legacy system represents significant investment of resources
- Often rule based and capture valuable domain features
- Isolate features and measure computing costs
- Use selected features in new models or process
- Related: Legacy, Adversarial

Internal Feedback

- Need a low risk way to test new models with live users
- Use your own product internally
- Give internal users a way to turn on new models, use the product, and give feedback
- Also use to develop training data
- Related: Bathwater, Follow The Crowd

Follow The Crowd

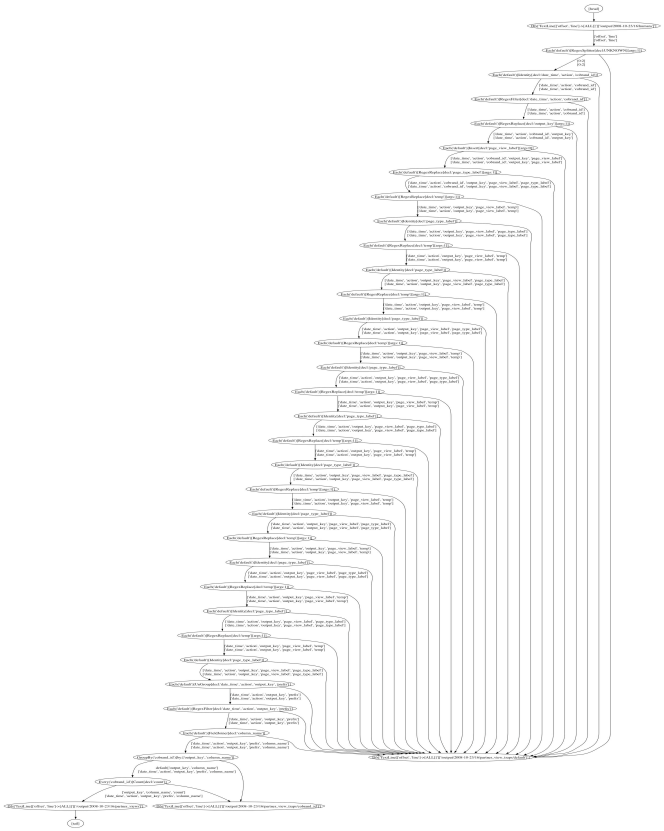
- Insufficient training or validation data for nobody to help
- Amazon's Mechanical Turk too low level
- Use a service such as Dolores Labs founded by machine learning researchers
- Labeling costs down to $0.05/label (source: http://doloreslabs.com)
- Related: Internal Feedback, Bathwater

Bathwater

- "Don't throw the baby out with the bathwater …"
- Subjective tasks can lead to "ML doesn't work" blanket rejection
- Isolate system elements that may be too subjective for ML and use human judgments
- Follow the Crowd (Crowd Sourcing)
- Related: Internal Feedback, Tiers

Pipeline

- A mix of computing and human processing steps need to be applied in a sequence
- Organize as a pipeline and monitor the workflow
- Individual cases can be teed off from the flow for different processing, etc.
- Related: Workflow, Handshake



Handshake or "Hand Buzzer"

- Your system depends on inputs delivered outside of the normal release process
- Create a "handshake" normalization process
- Release handshake process as software associated with input and version
- Regularly check for significant changes and send ALERTS

Replay

- Need a way to test models on operational data
- Invest in a batch test framework
- Example: web search replay query logs and look at changes in rank of clicked documents
- Example: recommender systems
- Example: messaging inbox replay

Tiers

- Processing or scoring elements have widely varying costs
- Often feature inputs or processing steps have orders of magnitude variation in computing cost or editorial costs
- Build models for each tier and only pass cases on to next tier if necessary
- Related: Thin Line, Pipeline

Long Goodbye

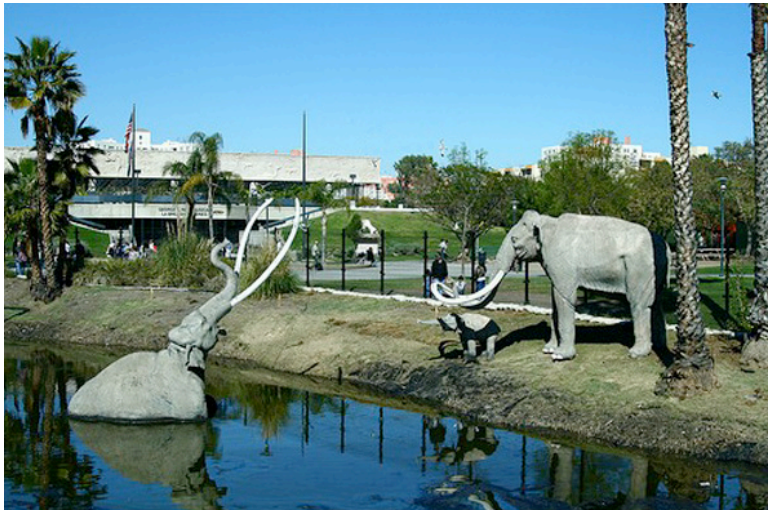
- Some decision classes have unacceptable risk or "loss"
- Isolate the high risk classes but don't remove from system entirely

- Example: quarantine or Bulk mail folders in email to keep false positives safe
- Delay rather than "reject" -- send uncertain cases to more costly processing steps rather than reject

Honey Trap

- New data streams are available for testing classifiers but data is unlabeled
- Isolate streams that are likely to be of one class or another
- Example: dead domains become almost entirely dominated by spam traffic
- (TN) Use to collect examples from examples with unknown labels like click fraud

Tar Pit



- System needs to identify bad entities but cost to register new ones is cheap
- Don't reject, delete, or notify bad actors
- Slows down adversary's evolution
- Example: slow down email messaging for low reputation IP addresses
- Related: Honey Trap, Adversarial

Example: Honey Trap + Tar Pit?

GetACoder.com

Home | My Account | Post Project | Browse Projects | RSS Feeds New!

Bug Finder

Budget: $ 300-1000
Status: Open for Bidding (30 days left)
Project Creator: AlanT pm
Rating: (No Feedback Yet)
Required Skills: Programming, Testing / Quality Assurance
Attached Files: (None)

NEW SEND DISCUSS CONTACT OPTIONS

Description
The purpose of this project is to create a debugger program. This program will take as input the source code another program, and will analyze that other program and determine if it will run to completion, or have an error, or go into an infinite loop.

To state that another way, given a function f and input x, determine if f(x) will halt.

Reminder
You may not start working in this and any project before your bid is accepted. An

Bids Received (18) | Shortlist (0) | Declined Bids (0) | Average bid

Place Bid | Post Similar Project | Send Project | Message Board(3) | Contact AlanT

Remember that contacting the other party outside the site (by email, phone, etc.) on a activity for such infringements and can immediately expel transgressors on the spot, s report it. Thank you for your help.

nbsp;

pm Shortlist Decline Bid **kagtech**

We have gone through your project specifications; give us the opportun timeline and schedule. We have a top-notch team. We ensure that your Development. We are ready to start the work. Thanks & Warm Regards
Bid Time: Today 05:00

pm Shortlist **sabasak**

location | bid amount | feedback | delivery time

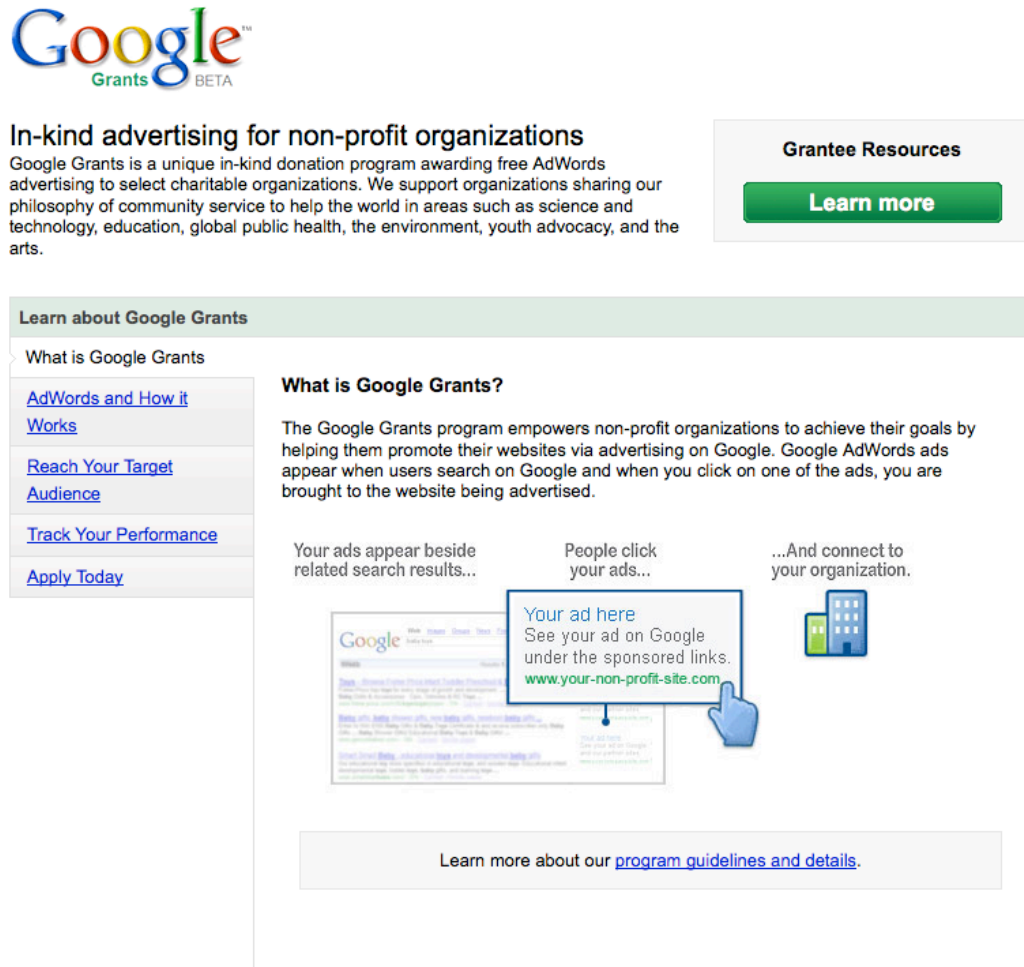
NEW SEND DISCUSS CONTACT OPTIONS

Description
The purpose of this project is to create a debugger program. This program will take as input the source code another program, and will analyze that other program and determine if it will run to completion, or have an error, or go into an infinite loop.

To state that another way, given a function f and input x, determine if f(x) will halt.

Giveaway

- Need low risk testing or new data
- Give away the service to non -customers
- Give away a related service (Google Analytics)
- Related: Honey Trap



Adversarial

- Adversaries are virulent and aggressive (email spam)
- Use regularization methods judiciously
- Parsimony can help make your adversaries' lives easier
- Test regularized and non-regularized models using Honey Trap
- (TN) Score by selecting from a set of models at random (mixed strategy?!)

Anti-Pattern Sampler

- Golden Sets (operational)
  - (+) Calibration
  - (-) Validation
- 80/20 (tactical)
  - (+) Design simplification
  - (-) "Good enough" can lose market share long term
- Executive Support (strategic)
  - (+) Resources
  - (-) Expectations
  - (-) Metric choices

Discussion

- Strategic
  - Thin Line
  - Legacy
  - Workflow
  - Bathwater
  - Giveaway
  - Contest (not presented)
- Operational
  - Honey Trap
  - Tar Pit
  - Handshake
  - Follow The Crowd
- Tactical
  - Pipeline
  - Tiers
  - Replay
  - Handshake
  - Long Goodbye
  - Shadow
  - Chop Shop
  - Adversarial
- Anti-Patterns
  - Golden Sets (operational)
  - 80/20 (tactical)
  - Executive Support (strategic)