Real-World Lessons in Machine Learning Applied to Spam Classification

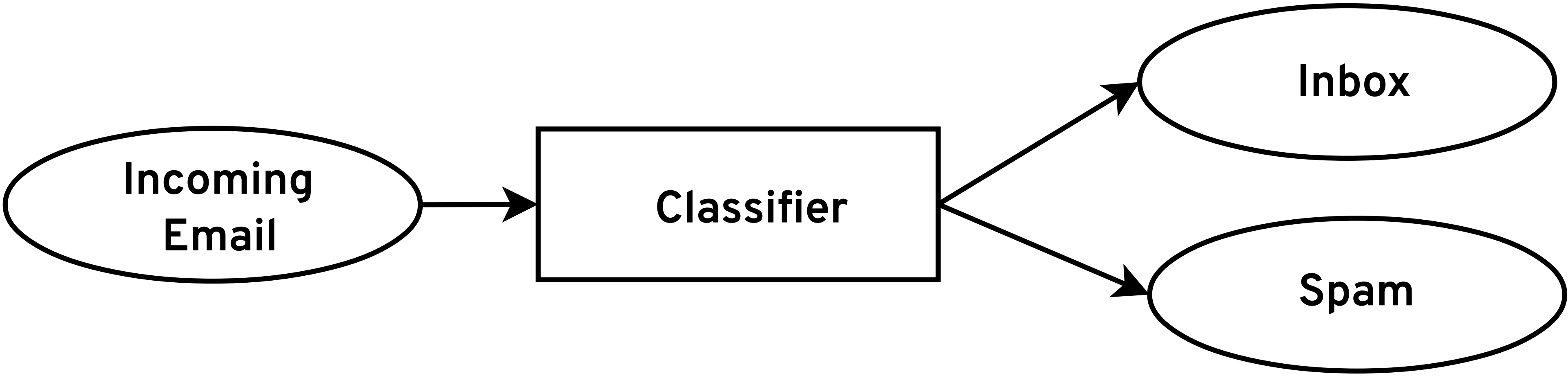
RJ Nowling

May 2, 2017

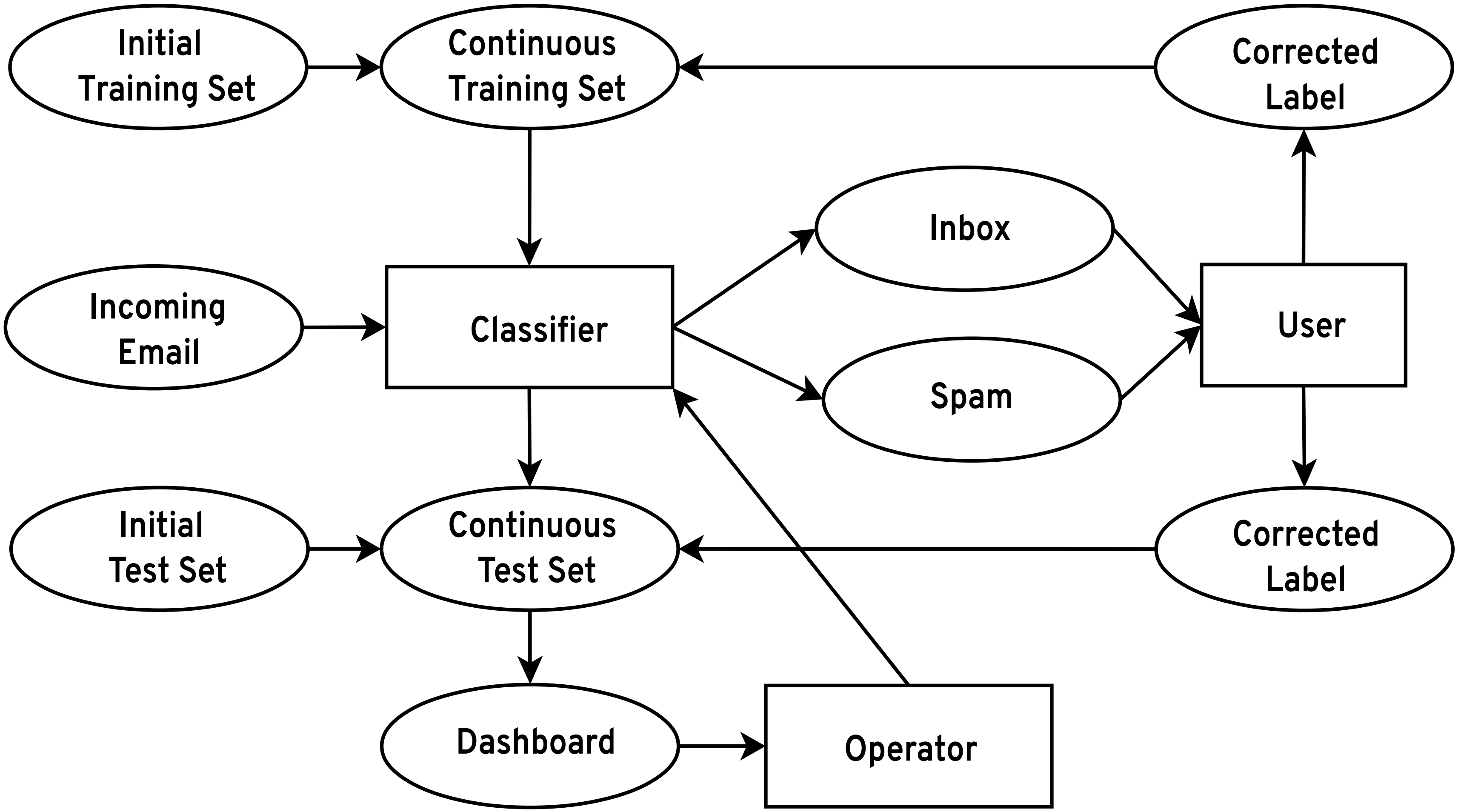
About Me

- Data Science Engineer at AdRoll
- Previously at Red Hat
- Apache BigTop Committer and PMC Member
- Ph.D. Computer Science & Engineering from University of Notre Dame
  - Research in machine learning for bioinformatics & differential equations for molecular simulation

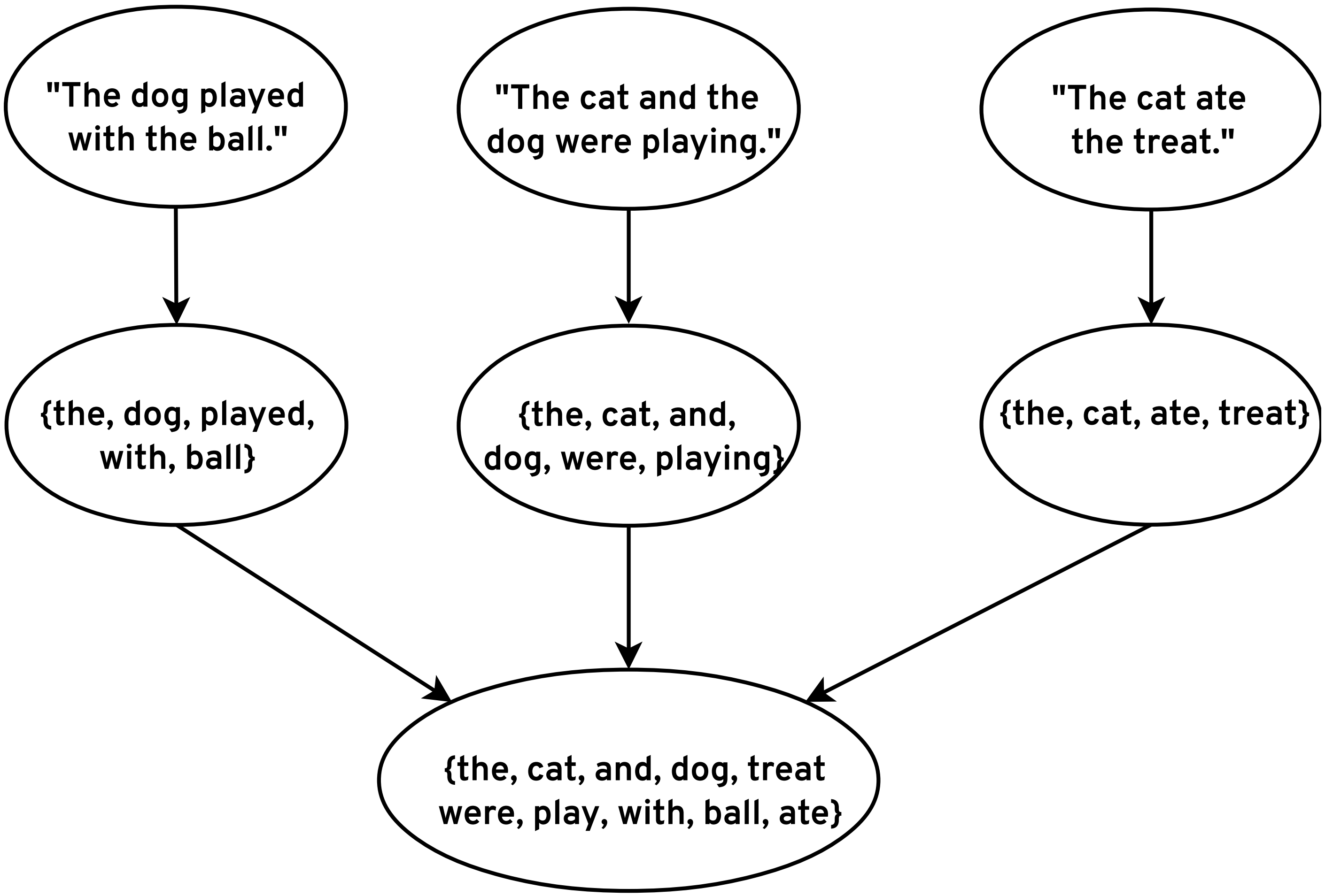
Separating Spam from Ham



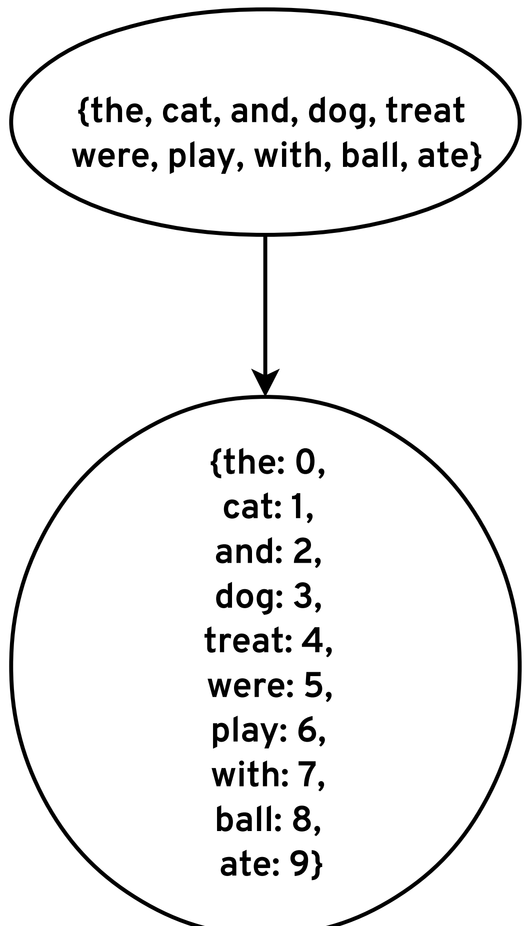
Spam Filter as a Service



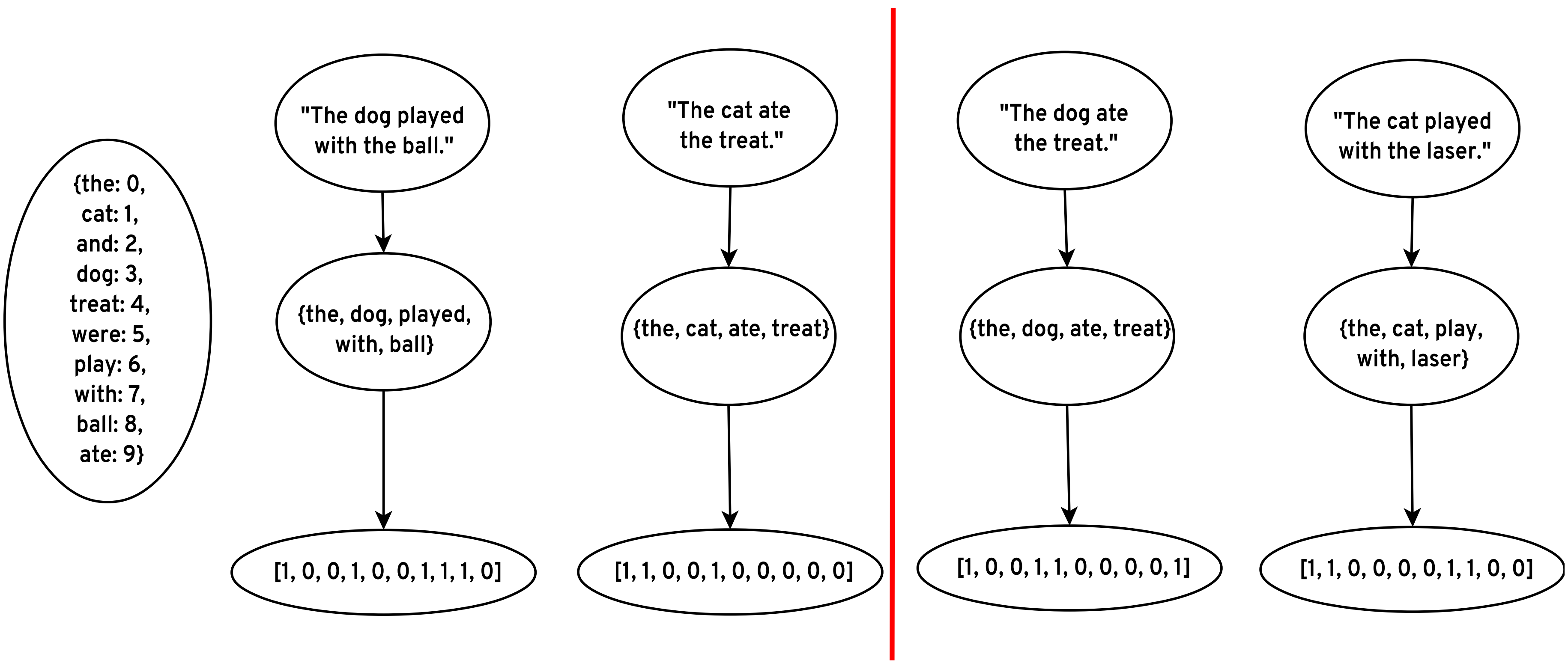
Extract Vocabulary



Map Words to Column Indices



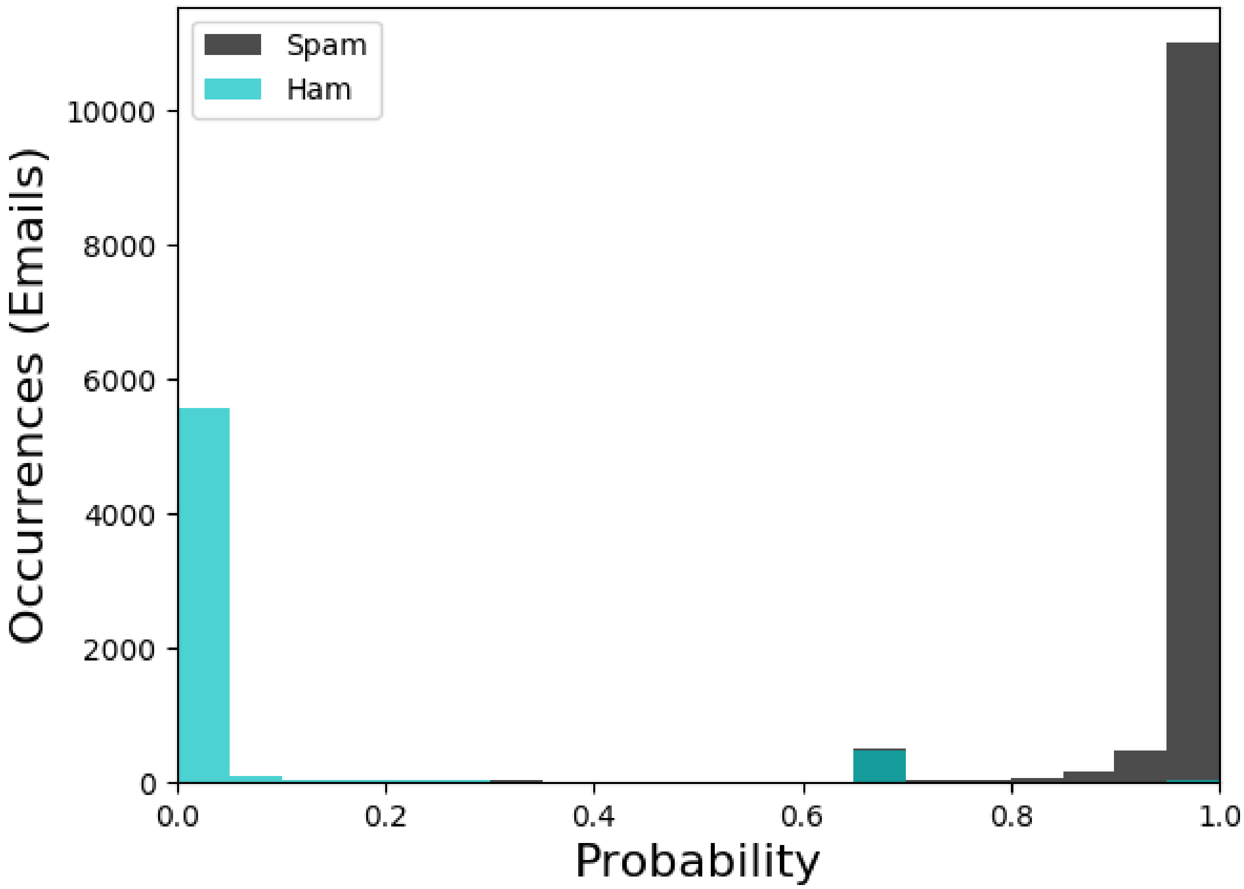
Encode Features



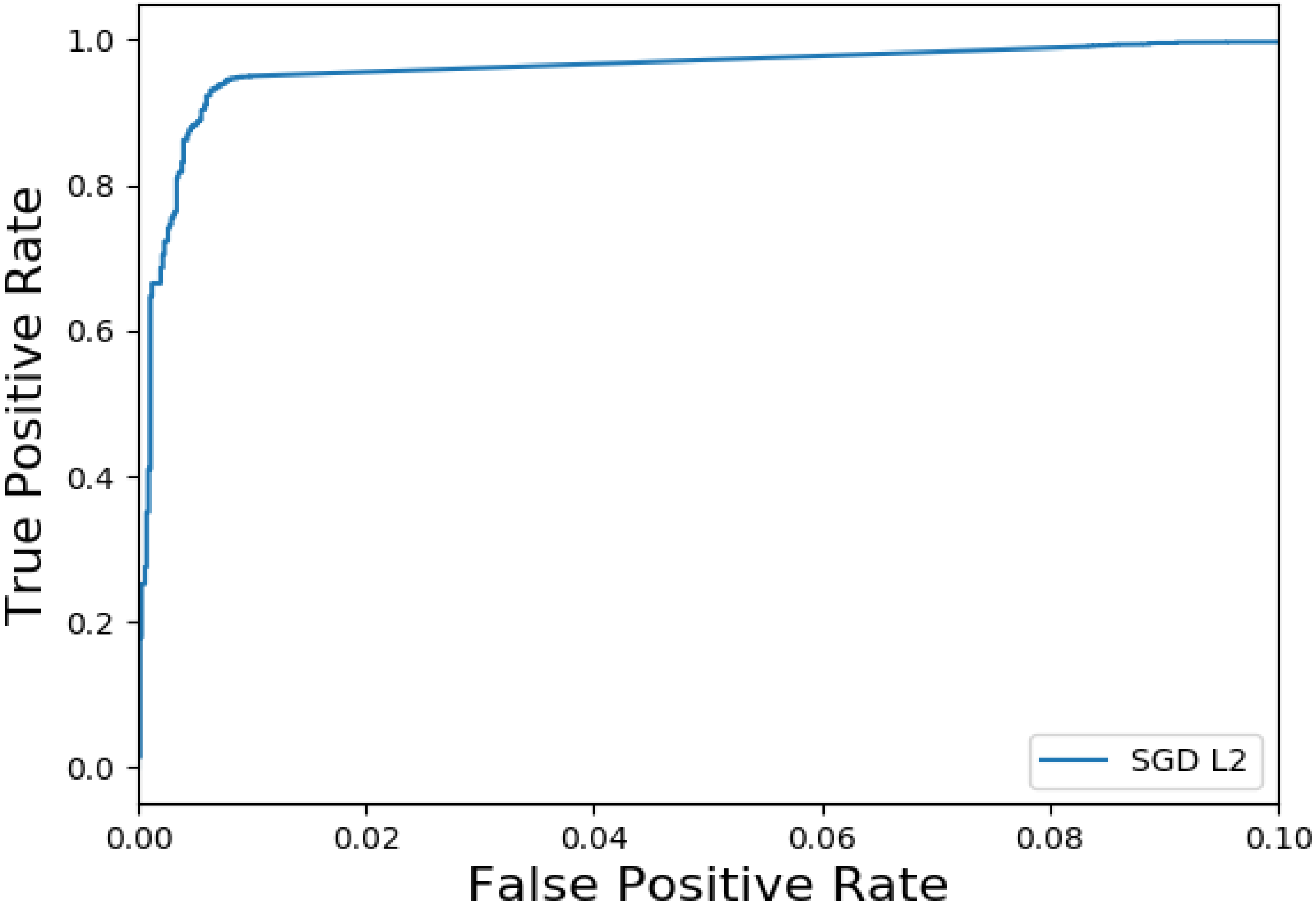
Logistic Regression

$$Pr(Y = 1|\mathbf{x}) = \frac{1}{1 + e^{-(\beta \cdot \mathbf{x} + \beta_0)}}$$

Predicted Probabilities

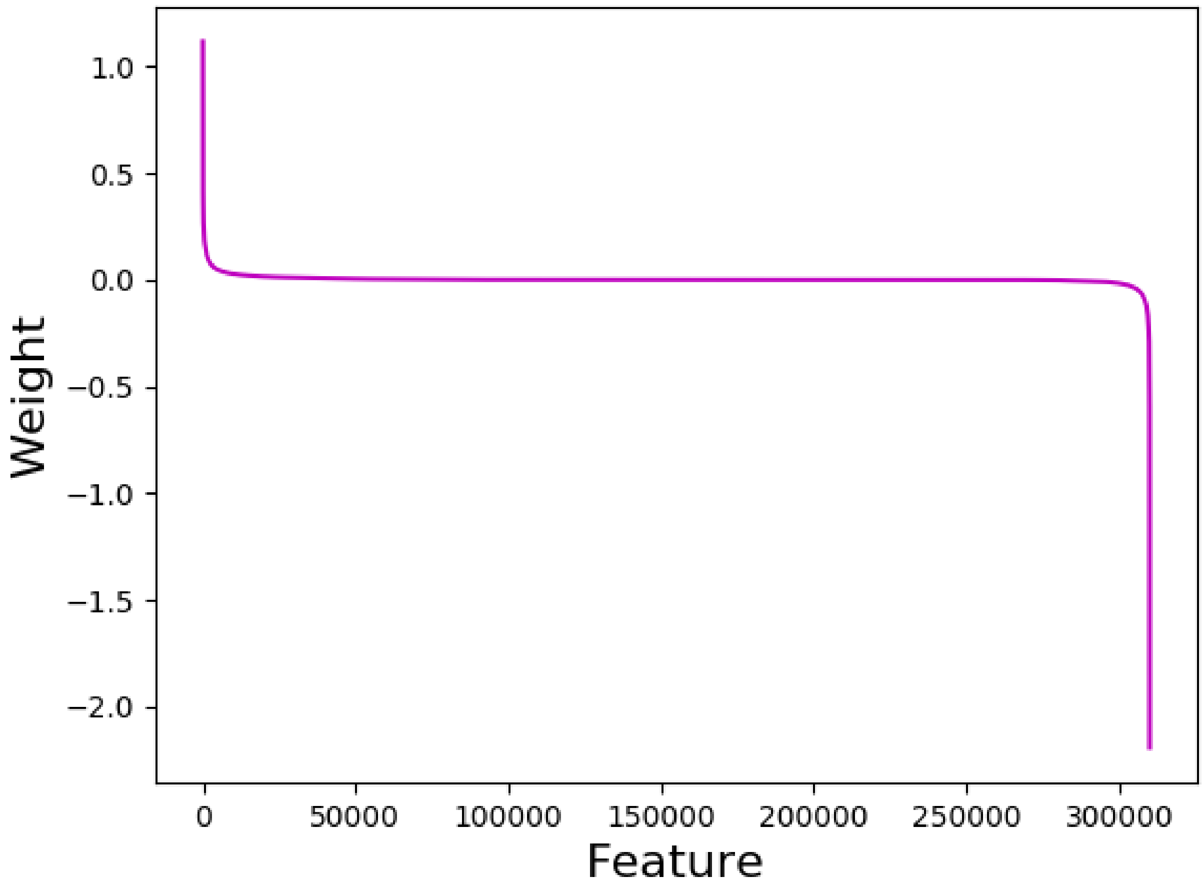


Evaluation



1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
SGD L2
0.00
0.02
0.04
0.06
0.08
0.10
False Positive Rate

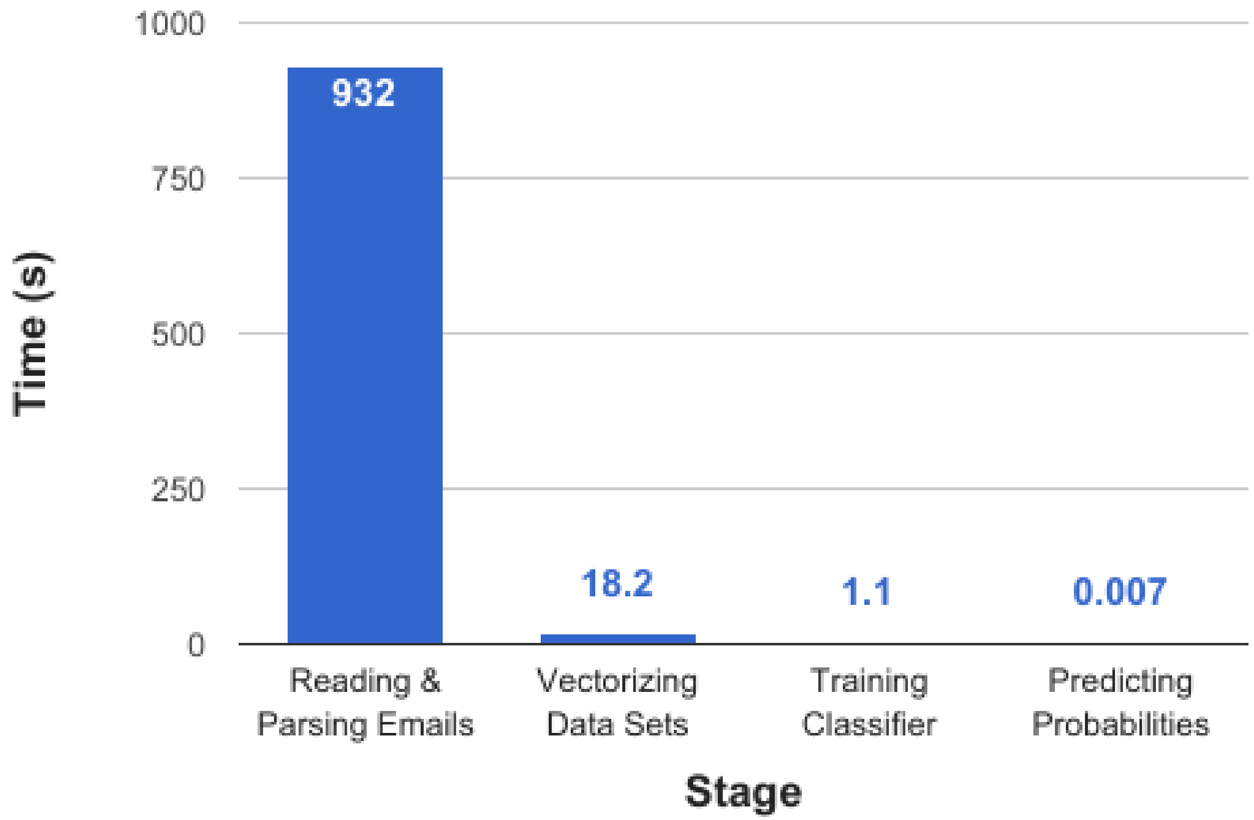
Feature Weights



Strongest Predictors

Weight	Word	Weight	Word
1.117	your	0.699	he
1.033	viagra	0.687	
1.017	productestpanel	0.686	properly
0.966	______	0.675	here
0.943	click	0.671	buy
0.902	price	0.653	net
0.884	you	0.648	http
0.803	symbol	0.642	viewing
0.774	hk	0.622	page
0.766	below	0.600	wkn

Timings

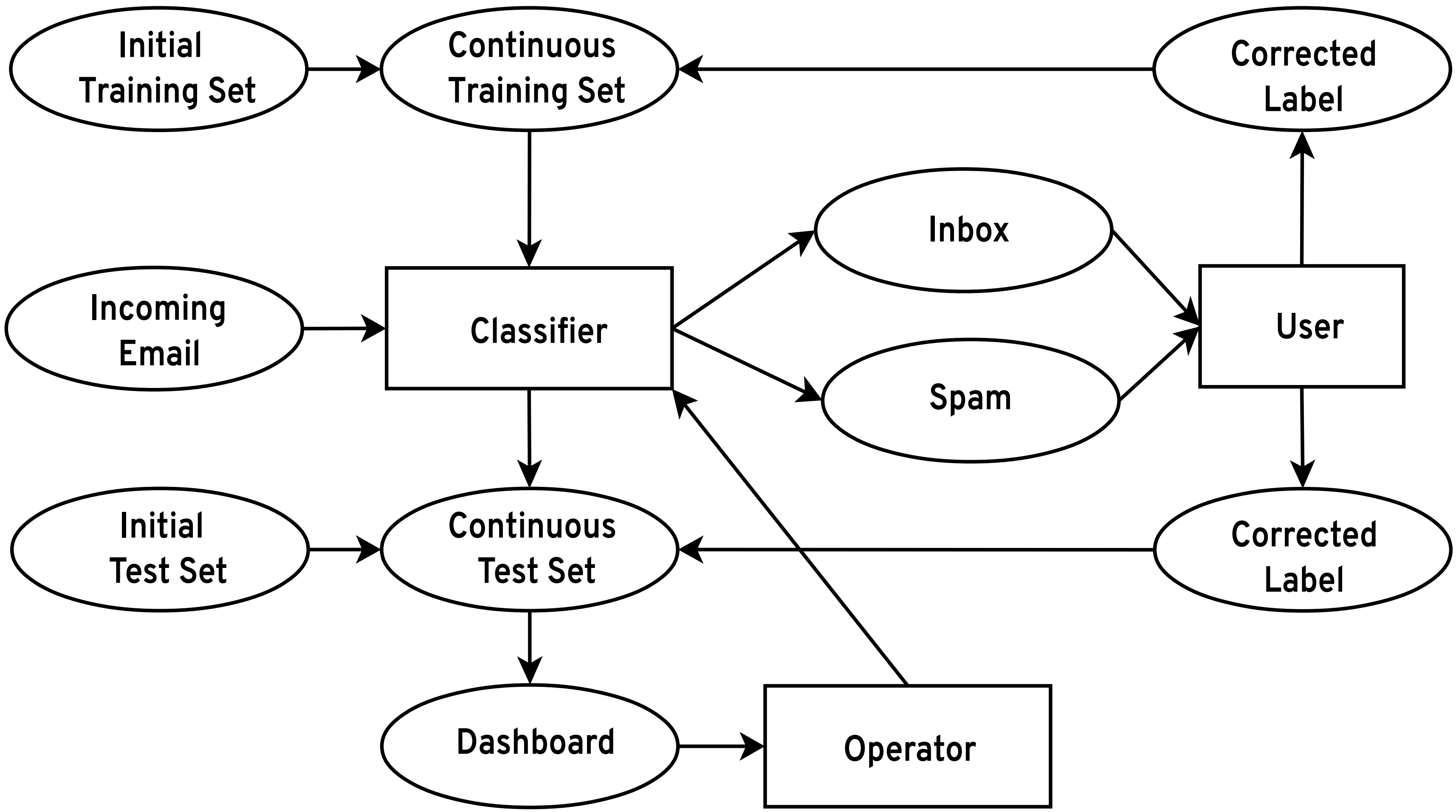


1000
750
500
250
0
Time (s)
932
18.2
1.1
0.007
Reading & Parsing Emails
Vectorizing Data Sets
Training Classifier
Predicting Probabilities
Stage

Story So Far

- Classifying emails as spam or not
- Need to encode document as numerical vectors – bag of words
- Very good accuracy (AUC of 99.5%)
- Interpreting model – spam words
- Reading and parsing emails is SLOW

Spam Filter as a Service



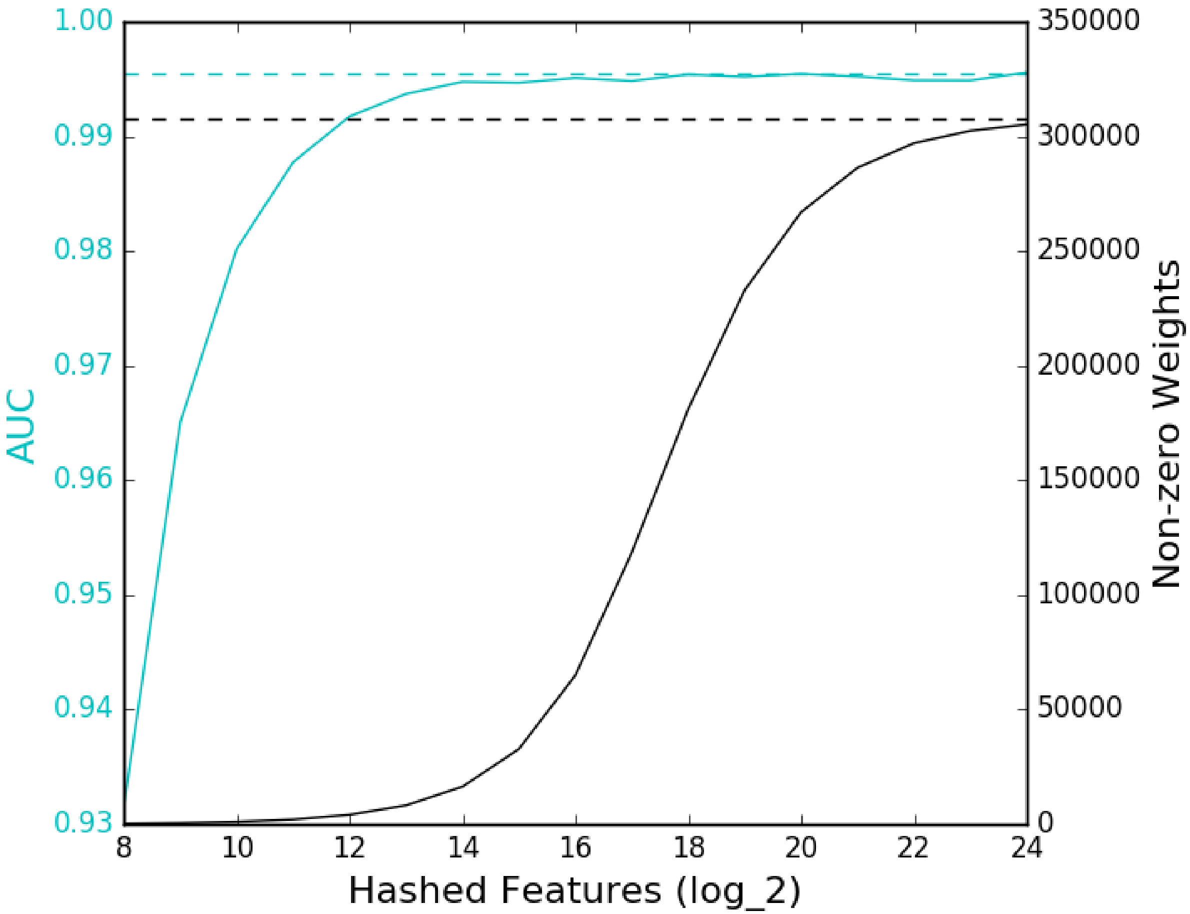
Hashing

- Hash(string) -> [0, 2^32 - 1]
- Hash(“dog”) -> 5
- Hash(“fog”) -> 8976234
- Hash(“cat”) -> 757676
- Uniformly distributed
- Avalanche effect: small change in input causes large change in output

Feature Hashing

```
features = np.zeros(n_features)
for word in document:
    idx = hash(word) % n_features
    features[idx] = 1
```

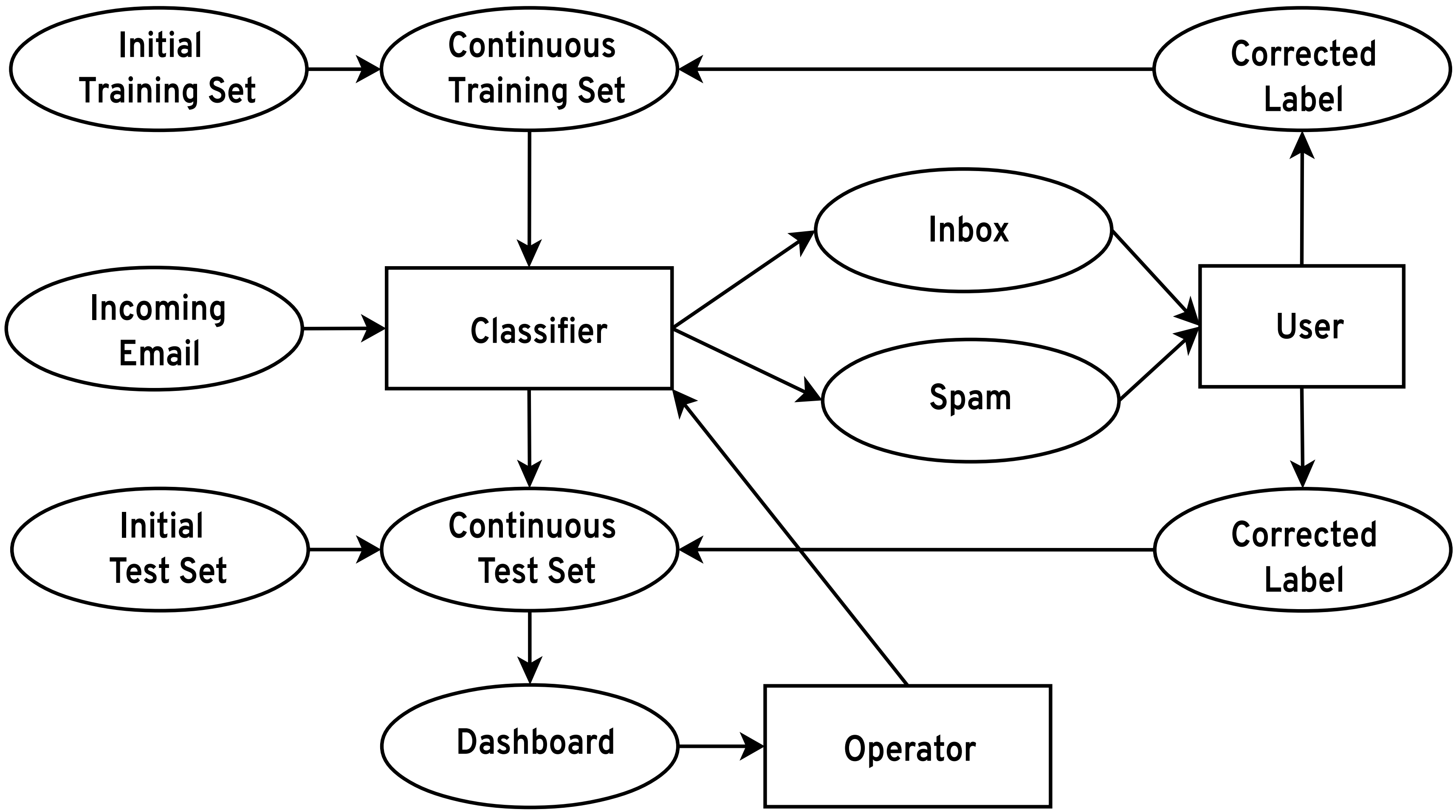
Accuracy and Collisions



Feature Hashing

- Fixed number of features (parameter)
  - Trade off between memory usage and accuracy
- Maps strings to indices based on the content of the string using hashing
- “Stateless” – nothing to update
- Include new vocabulary without re-vectorizing old training emails

Spam Filter as a Service



Online Learning

- For datasets too large to fit into memory, retraining models can be expensive (run-time, cost)
- Want to update models more frequently than permitted due to model training time
- Online learning: Update model only using new data points
- w/ Feature Hashing: New vocabulary is incorporated

VOWPAL WABBIT

https://github.com/JohnLangford/vowpal_wabbit

Food for Thought

Modeling and algorithm choices impact system design and operation.

And, system requirements guide our modeling and algorithm choices.

These are not independent.

Food for Thought

Models improve through new features, new algorithms, and more data.

Model performance can also deteriorate over time if new trends appear in data, but the model was trained on older, stale data.

Food for Thought

Feature engineering and algorithm design / implementation are (human) resource intensive and high variance.

But data collection and model updates can be automated and done continuously.

System continuously improves (freshens) itself for “free.”

Food for Thought

So, don't just think about building a model.

Think about designing systems that build models and your modeling / algorithm choices as part of the designing those systems.

Thanks!

Personalization

- Bob: Pharmaceutical representative
- Alice: Romance novel writer
- General model may predict incorrectly for these unique cases
- Want to personalize models per user

Per-User Model Challenges

- Many users = many models
  - Training time
  - Memory / storage requirements
- Very little feedback per user, no feedback from most users
- Solution: multi-task learning

Multi-task Learning

- We train a single model with user-specific features
- Accomplished via feature engineering
- Need feature hashing:
  - N users
  - M words
  - (N+1) M features vs fixed number of features

Feature Engineering

```
for user in users:
    for document in documents[user]:
        for word in document:
            general_idx = hash(word) % n_features
            features[general_idx] = 1
            user_idx = hash(user + “_” + word) \
                       % n_features
            features[user_idx] = 1
```