The Learning Behind Gmail Priority Inbox

Douglas Aberdeen **Ondrej Pacovsky** **Andrew Slater**
Google Inc.
Zurich, Switzerland
{daa,ondrej,aws}@google.com

Abstract

The Priority Inbox feature of Gmail ranks mail by the probability that the user will perform an action on that mail. Because "importance" is highly personal, we try to predict it by learning a per-user statistical model, updated as frequently as possible. This research note describes the challenges of online learning over millions of models, and the solutions adopted.

1 The Gmail Priority Inbox

Many Gmail users receive tens or hundreds of mails per day. The Priority Inbox attempts to alleviate such information overload by learning a per-user statistical model of importance, and ranking mail by how likely the user is to act on that mail. This is not a new problem [3, 4], however to do this at scale, performing real-time ranking and near-online updating of millions of models per day significantly complicates the problem. The challenges include inferring the importance of mail without explicit user labelling; finding learning methods that deal with non-stationary and noisy training data; constructing models that reduce training data requirements; storing and processing terabytes of per-user feature data; and finally, predicting in a distributed and fault tolerant way.

While ideas were borrowed from the application of ML in Gmail spam detection [6], importance ranking is harder as users disagree on what is important, requiring a high degree of personalization. The result is one of the largest and most user facing applications of ML at Google.

2 The Learning Problem

2.1 Features

There are many hundred features falling into a few categories. *Social features* are based on the degree of interaction between sender and recipient, e.g. the percentage of a sender's mail that is read by the recipient. *Content features* attempt to identify headers and recent terms that are highly correlated with the recipient acting (or not) on the mail, e.g. the presence of a recent term in the subject. Recent user terms are discovered as a pre-processing step prior to learning. *Thread features* note the user's interaction with the thread so far, e.g. if a user began a thread. *Label features* examine the labels that the user applies to mail using filters. We calculate feature values during ranking and we temporarily store those values for later learning. Continuous features are automatically partitioned into binary features using a simple ID3 style algorithm on the histogram of the feature values.

2.2 Importance Metric

A goal of Priority Inbox is to rank without explicit labelling from the user, allowing the system to work "out-of-the-box". Importance ground truth is based on how the user interacts with a mail after delivery. Our goal is to predict the probability that the user will interact with the mail within

T seconds of delivery, providing the rank of the mail. Informally, we predict $p = \Pr(a \in A, t \in (T_{min}, T_{max})|\mathbf{f}, s)$; where a is the action performed on the mail, A is the set of actions denoting importance (e.g., opens, replies, manual corrections), t is the delay between delivery and the action, $\mathbf{f}$ is the vector of features, and s indicates that user has had an opportunity to see the mail.

Imposing T_{min} is necessary to give users an opportunity to react to new mail, but is also constrained by how frequently we can update models. It is less than 24 hours. Imposing T_{max} bounds the interactions we need to consider and is a function of the resources available to store and process mail features. It is measured in days. A consequence is that users with interaction periods greater than T_{max} will not generate training data. To summarize, the prediction error is:

$$e = \begin{cases} 0 & \text{if } \neg s \vee t \notin (T_{min}, T_{max}) \\ 1 - p & \text{if } a \in A; \\ -p & \text{otherwise.} \end{cases}$$

2.3 Models

We use simple linear logistic regression models to keep learning and prediction scalable. A glut of data exists for learning a global model, but a dearth of data exists for learning personalized user models. We use a simple form of transfer learning [5] where the final prediction is the sum of the global model and the user model log odds (Fig. 1):

$$s = \sum_{i=1}^{n} f_i g_i + \sum_{i=1}^{n+k} f_i w_i, \qquad p = \frac{1}{1 + \exp^{-s}}.$$

The number of features is denoted by n. We use k additional user specific features that are not present in the global model. The global model weights are $\mathbf{g}$ and are updated independently and held fixed during personal model updates. Thus, the personal model weights $\mathbf{w}$ only represent how *different* the user is from the global model of importance. This results in more compact user models and the ability to quickly adopt changes in the global model, e.g., when new features are added.

We perform online passive-aggressive updates [2] with the PA-II regression variant to combat the high degree of noise in the training set. Each mail is used once to update the global model and once to update the model for the recipient of the mail, e.g. the update for the i'th user model weight is

$$w_i \leftarrow w_i + f_i \frac{\text{sgn}(e) \max(|e| - \epsilon, 0)}{\|\mathbf{f}\|^2 + \frac{1}{2C}},$$

where e is the error, C is a regularization parameter that tunes the "aggressiveness" of the updates and ϵ is the hinge-loss tolerance, or the degree "passiveness". In practice, we abuse C by adjusting it per mail to represent our confidence in the label, e.g. a manual correction by a user is given a higher value of C. User models also have higher C than the global model, and new user models have higher values still to promote initial learning.

2.4 Ranking for Classification

We determine a per user threshold for s to classify each mail as important or not important. We treat the problem as ranking rather than classification because tuning the threshold quickly is *critical* for user perceived performance. It is difficult to algorithmically determine the threshold that will make a user happy. Opening a mail is a strong signal of importance for our metric (Sec. 2.2), but many users open a lot of mail that is "interesting" rather than "important". Also, unlike spam classification, users do not agree on the cost of a false positive versus a false negative. Our experience showed a huge variation between user preferences for volume of important mail, which can not be correlated with their actions. Thus, we need some manual intervention from users to tune their threshold. When a user marks messages in a consistent direction, we perform a real-time increment to their threshold.

3 Production

Scaling learning to millions of users is as difficult as tuning the algorithms for a single user. To store and serve models, and to collect mail training data, we make extensive use of bigtable [1], which combines features of a distributed file system with a database.

3.1 Prediction Time

Priority Inbox ranks mail at a rate far exceeding the capacity of a single machine. It is also difficult to predict the data center that will handle a user's Gmail account, so we must be able to score any user from any data center, without delaying mail delivery. A bigtable is used to globally replicate and serve models to dedicated ranking tasks. After feature extraction and scoring, another bigtable is used to log the features to be used for learning.

Logging data to bigtable provides real-time merging of the mail features with subsequent user actions by maintaining a record per user:message-id. Thus all the data required for a model update is located in the same record. If we were to instead append all features and actions to a file on disk as they occur, hundreds of machines for several hours would be needed simply to aggregate and sort the log entries by user. Bigtable shares those resources across many applications and provides real-time record merging, making the data available globally for learning within minutes.

3.2 Learning

Sharding learning is conceptually simple. Each core is responsible for updating a fraction of user models. The challenge is to feed data over the network at a rate that keeps the cores busy. Bigtable does a lot of this work by providing global access to sorted user:message-id records. It is tempting to simply fetch the user model, perform updates for each message record, and write back the model. Unfortunately, with many millions of users the penalty for individual model reads and writes over the network is prohibitive. It is necessary to batch user model reads and writes, loading as many models as possible into RAM. For efficiency, bigtable performs batch reads in *approximate* key order, allowing parallelism across the servers that hold the data. Since messages are keyed by user:message-id, messages may be occasionally out of user order.

To evaluate our implicit metric we need to know the last time a user was active in Gmail. We cannot determine this until all user:message-id records have been read because they are not in temporal order. This requires two passes over the email data held in bigtable, per shard of user models. The first pass computes the last action time and other statistics over the shard of users. The amount of data transferred is small so the first pass is fast. The second pass scans over all message feature data, performing updates. Finally, all the user models that have changed are batch written back to the bigtable. Thus, each available core is given a fraction of users by user id prefix and that fraction is further split into shards of users where all the models can be held in RAM simultaneously (Fig. 2). The end result is that we can process 35 users per second per core, on *off peak* non-dedicated desktop-like machines. Some users have thousands of updates, some have one. This is a significant resource saving over true online learning with 24/7 dedicated tasks.

3.3 Data Protection

All user data is analyzed and stored in accordance with Google's privacy policy, and the features extracted from messages are deleted after training. For debugging and tuning, the team members only examined the features and statistical models of their own accounts.

4 Results

Fig. 3 shows a typical histogram of log-odds scores for the global model, with green (light) buckets indicating important messages (based on our metric), and red (dark) buckets indicating unimportant messages. This demonstrates how logistic regression produces a smooth ranking function. Each bucket contains a ratio of important to unimportant which follows the log-odds curve.

Based on our implicit importance definition, our accuracy $(tp + tn)/messages$ is approximately $80 \pm 5\%$ on a control group. Presentation bias causes accuracy to be 2 or 3% higher for active Priority Inbox users. This figure is better than it seems. Due to threshold tuning, our false negative rate is 3 – 4 times the false positive rate. Users read mail they acknowledge is not important, thus many of our false negatives are correctly classified from the user's point of view. This illustrates the difficulty of determining importance implicitly, the level of noise in the data set, and the challenge of evaluating user perceived quality. Manual markings from users are valuable because they provide a true

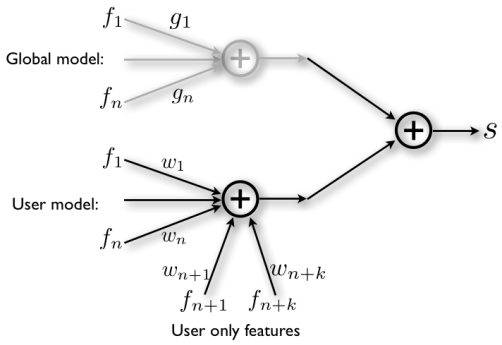

Figure 1: Adding personal and global model scores.

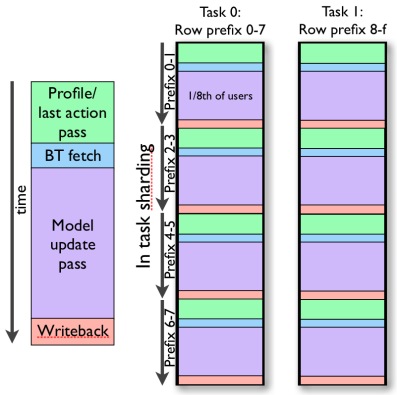

Figure 2: Sharding of user model learning.

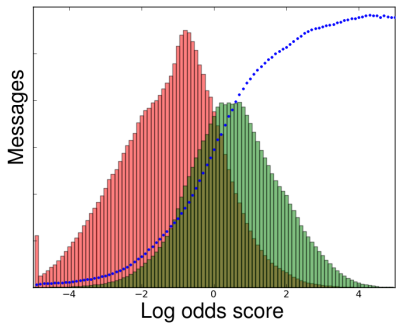

Figure 3: Score distribution for the global model. Dots show ratio of important to not important.

Combination	*Error*
Global model	45%
User models	38%
User models & thresholds	31%

Table 1: Error rates on user marked mail.

evaluation of importance, although they largely arise from classification errors and are hence biased. From a set of 160k such markings we computed the difference between applying only the global model versus personalized models and personalized models plus personalized thresholds (Tab. 1). The trend is that increased personalization significantly reduces mistakes.

The ultimate goal is to help Gmail users. We analyzed the time Google employees spent on email with and without Priority Inbox. Averaging over Googlers that receive similar volumes of mail, Priority Inbox users (approx. 2000 users) spent 6% less time reading mail overall, and 13% less time reading unimportant mail. They are also more confident to bulk archive or delete email.

References

[1] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: a distributed storage system for structured data. In *OSDI '06 Proceedings*, page 15, 2006.

[2] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai S. Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *JLMR*, 7:551–585, 2006.

[3] L. Dabbish, R. Kraut, S. Fussell, and S Kiesler. Understanding email use: Predicting action on a message. In *CHI 2005*, pages 691 – 700. ACM Press, 2005.

[4] Mark Dredze, Bill N. Schilit, and Peter Norvig. Suggesting email view filters for triage and search. In *In Proc. IJCAI'09*, pages 1414–1419, 2009.

[5] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010.

[6] Bradley Taylor. Sender reputation in a large webmail service. In *Third Conference on Email and Anti-Spam (CEAS 2006)*, 2006.