The University of Illinois' Graduate School of Library and Information Science at TREC 2012

Miles Efron, Jana Deisner, Peter Organisciak, Garrick Sherman, Ana Lucic
Graduate School of Library and Information Science
University of Illinois, Urbana-Champaign
501 E. Daniel St., Champaign, IL 61820
{mefron, jdeisner, organis2, gsherma2, alucic2}@illinois.edu

1 Introduction

The University of Illinois' Graduate School of Library and Information Science (uiucGSLIS) participated in TREC's microblog and knowledge base acceleration (KBA) tracks in 2012. Our high-level goals were two-fold:

- *Microblog*: Test a novel method for document ranking during real-time search.
- *KBA*: Compare methods of topic representation–particularly longitudinal adaptation of topic representation–for the KBA task.

Our document ranking in the microblog track is based on a behavioral metaphor. Given a query Q, we decompose Q into a set of imaginary saved searches S. Given an incoming document stream $\mathbf{D} = D_1, D_2, \ldots, D_N$, we ask: what is the probability that a document D is read, given the user's query and a rational allocation of attention over his saved searches?

Our KBA runs relied on the track's inherent temporality to induce and maintain expressive entity profiles during the Cumulative Citation Recommendation (CCR) task. Time influenced our approach in two ways. First, an initial entity model was built by analyzing that entity's Wikipedia edit history prior to the corpus start date. Each feature's initial probability was a function of its persistence over the edit history. Second, as our system iterated over the stream's nine-month window, an entity's model adapted in light of recently seen documents. Entities were represented as weighted feature sets using the Indri query language. Our main result hinged on the method for model adaptation over time. In adapting for temporal change, we used a monthly "memoryless" updating procedure, balancing a conservative approach to updating with more aggressive adaptation. Thus, at the end of each month of filtering, an entity's features were updated based on the month's stream. This update mixed the previous month's model with the current model; but all previous months were ignored. No new features were added; rather, we only re-estimated

each feature's probability within the model. This approach allowed a helpful degree of adaptability, while hedging against model drift. Of particular interest was the effect of model updating mechanisms. We used two very different approaches to adaptation. On the supplied training data, a Markov model-based technique strongly outperformed a simple linear interpolation. However, initial results suggest that in the long run, the formalism behind updating was less important than the more basic matter of assuring adequate flexibility by some means while retaining the model's pertinence to the entity it described.

Our system used the Indri search engine and API[1] for core indexing and data manipulation. Our retrieval models varied from task to task, as described below. Very little pre-processing was used in our experiments. We did not stem documents. We did no stopping at index time, though stoplists (described below) were used during construction of some baseline pseudo-relevance feedback models.

2 Microblog Track

We submitted runs for only one of the microblog track's two tasks: real-time ad hoc retrieval. This task required teams to retrieve documents posted to the Twitter microblog service[2] in response to a query issued at a particular time. Track organizers created 50 test topics and accumulated relevance judgments in a fashion similar to the standard TREC pooling method. Details of this process are available in the track overview paper.

We downloaded a version of the tweets2011 corpus on May 25-26, 2012 using the HTML scraping tools provided by the track organizers. This crawl yielded 11,908,900 indexable tweets with an HTTP status code of 200. These comprised our experimental corpus.

Our main goal was to capitalize on the essentially temporal character of people's interactions with Twitter. With this in mind, we formulated a retrieval model appropriate for inherently temporal information.

2.1 A Saved-Query Model for Real-Time Search

Our tweet ranking model is based on a behavioral metaphor. Assuming that a person has an information need $\mathcal{I}$ articulated by the query $Q = (q_1, \ldots, q_{n(Q)})$, we imagine Q as a set of saved searches $S = (s_1, s_2, \ldots, s_{|S|})$, where $|S|$ is the number of unique terms in Q. This fictional user then monitors incoming documents by checking tweets retrieved by his saved searches. However, a user only has a finite amount of attention to allocate to monitoring his saved search results. Thus, we score a document D on the probability that a user would read D given a *rational* allocation of attention over his $|S|$ saved searches. We denote this

[1] `http://lemurproject.org`

[2] `http://twitter.com`

as $P(\mathcal{D}|S)$: the probability that D is read given a set of saved searches S:

$$P(\mathcal{D}|S) = \sum_{s \in S} P(\mathcal{D}|s)P(s|S). \tag{1}$$

It is important to stress that the random variable $\mathcal{D}$ has a binary outcome: a document is either read or it is not. This is in opposition to any sort of generative model.

Microblog documents are scored according to Eq. 1 in our runs.

We can read the summation in Eq. 1 as being over "the probability that D is read given that we are scanning the results of saved search s times the probability that we choose to scan the results of s" for each $s \in S$.

To estimate $P(s|S)$ we assume that the searcher allocates his attention to each saved search s according to $P(s|Q)$, under the intuition that the query model expresses the searcher's topical priorities. Initially, then, we simply have the maximum likelihood estimate:

$$\hat{P}_0(s|S) = \frac{n(s, Q)}{n(Q)} \tag{2}$$

If we have a set of (pseudo-) relevant documents $R = (D_{R1}, D_{R2}, \ldots, D_{Rk})$ we have:

$$\hat{P}_R(s|S) = \frac{n(s, R)}{n(R)} \tag{3}$$

where $n(R)$ is the number of tokens in all k feedback documents, and $n(s, R)$ is the frequency of s in all documents comprising R. Given a mixing parameter $\alpha \in [0, 1]$, we have the feedback distribution:

$$\hat{P}_F(s|S) = \alpha \hat{P}_0(s|S) + (1 - \alpha)\hat{P}_R(s|S). \tag{4}$$

Throughout the following discussion we set $k = 20$ and $\alpha = 0.5$. We also restrict the number of terms added to the collection of saved searches via feedback to the the 20 terms with highest $\hat{P}_R(s|S)$.

The more difficult task lies in estimating the user's probability of reading a particular document retrieved by the saved search s. The key to our approach here lies in the problem of information overload.

Creating a saved search for a very common word such as *the* will of course lead to high recall–many relevant documents contain this term. But over time, the results of this search would be unmanageable. We assume that users have finite time to allocate to reading the results of their searches. If a search accumulates many documents, it carries a risk that the user will fail to read a relevant document due to the scarcity of attention. As the frequency of s in the document stream increases, the probability of reading a given document in its collected results decreases. Intuitively, $P(\mathcal{D}|s)$ monotonically decreases with the frequency of s.

To capture this intuition, we imagine the following. At time t the user reviews the results obtained by the saved search s. This consists of n_s documents. While one could conceivably make various assumptions about the ordering of these n_s documents for presentation to the user, we assume no particular order (i.e. a simple set-based retrieval is at work). Below we discuss the implications of this choice. The user begins to read these n_s documents. After each one, he chooses either to continue onto the next document or to quit. In the interest of simplicity, we assume that λ, the probability of quitting, is constant. This leads to an exponential distribution with rate parameter λ. The probability of reading at least m documents is thus $e^{-\lambda m}$.

Let us consider a particular document D among all n_s retrieved documents. We wish to know the probability that the user will read D (the event $\mathcal{D}$). If D is first in the user's "in-box" we have $P(\mathcal{D}) = e^{-\lambda 1}$. If λ is small, this is a very likely event. But if D is buried deep among the n_s documents, requiring the user to read, say, 1000 documents before he finds it, the probability of reading d is very small.

Let $E(r_D)$ be the expected rank of document D in the results accumulated for s at time t. The probability of D being read given a result set of n_s documents is:

$$P(\mathcal{D}|s) = e^{-\lambda \cdot E(r_D)}. \tag{5}$$

If we imagine that the order of documents in the results for s follows no particular ordering (i.e. we imagine a simple set-based search), then $E(r_D) = \frac{1}{2}s_n$. This amounts to having no influence of term frequency in our document scoring. However, we could easily introduce a local term frequency factor into the ranking by assuming that $E(r_D)$ follows some function that is increasing on the frequency of s in D. Alternatively, we might imagine a temporal ordering, with newer documents promoted via and expectation using a distribution favoring recently published information. However, we leave that for future work.

In this work, we assumed a uniform distribution for document ranks in saved search results, which means that we relied only on binary features. As in the binary independence model, we effectively treated each query term's occurrence in a document as a Bernoulli trial.

The choice of λ in Eq. 5 governs the importance of an IDF-like factor in document ranking. Setting $\lambda = 0$ leads document frequency to play no role in the utility of a word. On the other hand, large values of λ lead to a very strong penalty for common words.

In our work, based on ad hoc tuning using training queries, we set $\lambda = 0.001$ ***in our experiments.***

In many ways, the "attention model" given in Eq. 1 is analogous to simple TF-IDF ranking. Its first factor is a TF weight, the second factor expresses the frequency of each term in the query (or feedback documents), and the final term is similar to IDF. However, our model offers several points of flexibility:

1. Tunable IDF influence via selection of λ.

2. Arbitrary functional form for IDF influence (e.g. Weibull distribution for quit probability instead of simple exponential).

3. Ability to account for arbitrary document features (query-dependent or independent) by the expectation in Eq. 5.

2.2 Submitted Runs

We submitted four runs, all based on the model described in the previous section, with varying levels of additional sophistication. Except for uiucGSLIS01, none of our runs relied on external or future evidence. In uiucGSLIS01, we used external evidence in the form of the vocabulary from the TREC AP 88-89 corpus to assess the degree to which a tweet's language conformed to conventional English. In almost every sense, like our other runs, uiucGSLIS01 ignored future evidence. However, the weights in the learned model were trained on the topics from the 2011 microblog track. Some of these topics have query times posted after query times for the 2012 topics, so in a strict sense we were learning from future data in this case.

2.2.1 uiucGSLIS04

This run used the retrieval model described in Section 2.1. It used no feedback and was intended to serve as a simple baseline.

2.2.2 uiucGSLIS02

This run used the retrieval model described in Section 2.1. We used pseudo-relevance feedback (20 documents, 20 terms, with the feedback model interpolated with the original query via a mixing parameter of 0.5). An important point is that our relevance feedback method used no stoplist (unlike the relevance model described below).

2.2.3 uiucGSLIS03

This run used the retrieval model described in Section 2.1. We used pseudo-relevance feedback (20 documents, 20 terms, with the feedback model interpolated with the original query via a mixing parameter of 0.5).

Additionally, each document's score in this run was multiplied by an exponential factor calculated in the method described in [1].

2.2.4 uiucGSLIS01

This run was intended to show our most competitive effort. The ranking model described above was put into a larger learning to rank framework. The features used in our ranking were largely due to observations from successes of last year's microblog participants.

Table 1: Feature weights in ranking model for uiucGSLIS1 run.

Feature	Weight
Textual similarity (with pseudo-relevance feedback)	1.0
log length (in characters)	0.075
Has link (boolean)	0.399
Percent of vocab in TREC AP corpus	0.676
Recency score	1.675
Has hashtag (boolean)	-0.025
Has mention (boolean)	-0.55
Percent of vocab in handmade list of "spam" words	-0.675

The weights shown in Table 1 we learned by a simple coordinate ascent, using the 2011 microblog topics for training, with MAP as the objective function. The main surprise in these weights in the influence given to document recency. The recency score was a simple exponential decay function of the age of the document with respect to the query; the rate parameter on the exponential was $\lambda = 0.01$, with time measured in days. Previous research has shown that many of the 2011 topics were in essence "recency queries," with a preponderance of relevant documents posted near query time. This led to the strong temporal influence seen here. We found, however, that the 2012 topics were much less sensitive to time.

2.3 Microblog Ad Hoc Empirical Results

Table 2: Initial Report of Microblog Ad Hoc Experimental Results.

Run	MAP	Rprec	P30
median	0.1487	0.1869	0.1808
uiucGSLIS01	0.1829	0.2080	0.2186
uiucGSLIS02	0.1751	0.2020	0.1972
uiucGSLIS03	0.1717	0.1959	0.1983
uiucGSLIS04	0.1259	0.1482	0.1599
QL	0.1333	0.1602	0.1627
Rel Model	0.1558	0.1912	0.1684

The row in Table 2 labeled *median* shows the median effectiveness scores aggregated over all official runs. The rows labeled *QL* and *Rel Model* were not submitted, but are shown

as a baseline. The *QL* row gives results using the standard query likelihood model; this is intended as a point of comparison with uiucGSLIS04. The row labeled *Rel Model* shows results obtained from applying feedback via relevance models with feedback configurations as in uiucGSLIS02, with a strong stoplist applied.

Not surprisingly, uiucGSLIS01–a many-featured model–performed well. However, its advantage over uiucGSLIS02 and 03 is actually slim. In comparison to uiucGSLIS04 we can see that the lion's share of effectiveness in these runs is due to the influence of relevance feedback. Except for uiucGSLIS04, all of our runs outperform the population median.

Comparing uiucGSLIS with QL shows that the attention model ranking method is performing near the quality of a well-tested language modeling approach. More intensive parameterization of the attention model may well fill this gap, especially insofar as the difference in effectiveness between these runs is not statistically significant. Additionally, relaxing our reliance on binary feature representation may help improve the settings of uiucGSLIS04.

We were gratified by the comparison of uiucGSLIS02 and the *Rel Model* run. This comparison showed feedback in the attention model outperforming the more established feedback method. Additionally, while obtaining strong scores with relevance models required the use of an aggressive stoplist, feedback in our approach does not use a stoplist.

3 TREC 2012 KBA Track

The Cumulative Citation Recommendation (CCR) task in this year's knowledge base acceleration (KBA) track challenged teams with the following scenario: given a Wikipedia entity $\mathcal{E}$ represented by a Wikipedia node E, monitor an incoming document stream, signaling to the editors of E when an "edit-worthy" document is seen. Track organizers identified 29 target entities. Teams monitored a stream of timestamped documents–mostly news articles and blog posts–spanning approximately a nine-month window.

3.1 Data

We used only the "cleansed" subset of the KBA document stream. The cleansed subset was generated by track organizers by limiting documents to those with a "reasonable" chance of being in English. We chose to work with the cleansed data for two reasons: 1) Its relatively small volume reduced engineering challenges, and 2) all relevance judgments were limited to documents in the cleansed subset. Track organizers also made named entity (NER) data for each document available to participants. However, NER information was not central to our interests so we did not use this information.

Relevance assessments over the 29 target entities took on one of four integer values corresponding to four categories. We itemize them in Table 3. The rightmost column of Table 3 stems from our interpretation of what constituted relevance. We limited relevant

judgments to central documents. In our discussion below, we also report results using NDCG, which used the raw assessment scores.

Table 3: TREC KBA Document-Entity Annotations with Associated Relevance Interpretations.

Score	Label	Our interpretation
-1	garbage	Not relevant
0	neutral	Not relevant
1	relevant	Not relevant in our evaluations
2	central	Relevant in our evaluations

3.2 Motivation and Approach

Because this was the first year that KBA ran at TREC, the best way to consider the task was an open question. For instance, participants could think of the system as a binary classification problem. Alternatively, we could approach it as a document ranking problem. We chose the latter approach, aiming to score each document with a real-valued number instead of a binary $+/-$ decision. This outcome, we reasoned, would allow a Wikipedia editor to browse documents in decreasing order of predicted relevance, while also admitting a simple thresholding if she wished to see only those documents judged to be either "relevant," "central" or both. Lastly, all system parameters were set by experimentation on the supplied training data. No rigorous optimization was done.

The main focus of our work was entity representation. We explored the ability of the tasks' inherent temporality to give information that would allow the profile for $\mathcal{E}$ to evolve over time. We also experimented with the value of the edit history of the Wikipedia page for $\mathcal{E}$ in learning a topic representation.

Each entity $\mathcal{E}_i$ was represented by an indri query Q_i. At the outset of a run, we began with the *initial query* Q_i^0. At any time t in the iteration over the document stream, we have the *current query* Q_i^t. We will return to our methods for estimating Q_i^0 and Q_i^t, but first there are a few details to note.

Taking a very conservative approach, we only returned documents that contained an exact match on the canonical entity name (i.e. the name represented in the Wikipedia URL) for $\mathcal{E}$ or a very close match. For instance, the query for entity `Aharon_Barak` contained the filtering statement `#scoreif(#1(Aharon Barak))`. However, this stricture ignored disambiguation information. So, the entities `Basic_Element_(company)` and `Basic_Element_(music_group)` both filtered on `#scoreif(#1(Basic Element))`. Similarly, the URL `Frederick_M._Lawrence` contained the filter `#scoreif(#syn(#1(Frederick Lawrence) #1(Frederick M Lawrence)))`.

Our queries also combined several types of information. First, documents were promoted based on the log of their inbound link count plus 1. This factor was implemented as a document prior (the log-counts were divided by the sum of all log-counts) to achieve a prior distribution over documents. Second, the indri query described below were applied to both the text of documents and their resolved URLs. The weight of evidence in URL text was 2.0 versus 1.0 in the document text. Lastly, in each indri query, the learned model was weighted 0.35 while the name of the entity was weighted 0.65.

3.2.1 Initial Query Model Estimation

For an entity $\mathcal{E}$ we created an initial query Q^0 by analyzing the edit history of $\mathcal{E}$. We represent this history by the sequence of documents $H = E_1, E_2, \ldots, E_k$ where E_k is the last version of $\mathcal{E}$'s Wikipedia entry before the start of the KBA document stream. Iterating over this history we extracted a set of features and weights used to represent $\mathcal{E}$.

Extracted features were any wikiText that appeared as a link using the **[[.]]** ***notation. These features were treated as quoted phrases.***

For each feature f_i we estimated $P(f_i|H)$. To estimate these probabilities we simply calculated:

$$P(f_i|H) = \frac{1}{k} \sum_{E \in H} \hat{P}_{ml}(f_i|E). \tag{6}$$

As a point of comparison, we built a set of queries whose weights ignored edit history in favor a estimating probabilities only from the most recent version E_k. However, this condition did not inform any of our official runs.

3.2.2 Query Model Adaptation

A key factor in KBA is the change that information about an entity $\mathcal{E}$ can undergo over time. We reasoned that over a nine-month window, the initial query model Q^0 would become "stale." To remedy this, we desired some sort of adaptation over our models. However, we did not assume any explicit feedback. Likewise, we speculated that it would be risky to let incoming documents influence the model too much, in which case the risk of topic drift would become large.

Our query model adaptation utilized a month-by-month, memoryless updating procedure. That is, at the beginning to the KBA process we used Q^0 to score incoming documents. After one month, the system gathered the highest scoring $n = 20$ documents from the initial month, updated the model based on these results (as described below) and then processed the next month's documents. After each month, the most recently accumulated documents were used to update the model.

To avoid query drift, we took a conservative approach and allowed models to change only in a re-weighting of the originally observed features. i.e. No

features were added to the model, although as described below, some features' weights went to zero.

The model was memoryless in the sense that at time t (for $t > 0$), the parameters of the model were fully specified by the results from month $t-1$ and t. i.e. Months $t-2, t-3$... exerted no influence. This choice was made in order to allow the model to avoid stagnation. Thus at time $t > 0$ we had the model

$$Q^t = \phi(Q^{t-1}, R_t) \tag{7}$$

where R_t is the set of pseudo-relevant documents retrieved during month $t-1$ by query Q^{t-1}. The two runs described below vary only in their specification of the updating function $\phi(\cdot, \cdot)$.

3.3 Submitted Runs

3.3.1 gslis_adaptive

This run generated a query Q^t based on the previous month's query Q^{t-1} and the pseudo-relevant documents R via a simple linear combination:

$$P_a(f|Q^t) = \alpha P(f|Q^{t-1}) + (1-\alpha)\hat{P}_{ml}(f|R) \tag{8}$$

for a mixing parameter $\alpha \in [0, 1]$. We simply set $\alpha = 0.5$..

3.3.2 gslis_mult

This run generated a query Q^t based on the previous month's query Q^{t-1} and the pseudo-relevant documents R using a log-linear model:

$$P_m(f|Q^t) = [\lambda \cdot \frac{1}{|Q|} + P(f|Q^{t-1})] \cdot [(1 - \lambda \cdot \frac{1}{|Q|}) + \hat{P}_{ml}(f|R)] \tag{9}$$

for a smoothing parameter $\lambda = 0.01$.

3.4 KBA Empirical Results

Table 4 summarizes the outcome of our runs in comparison to two participant-wide aggregates– mean and median F1. Two results are clear from these data:

1. Neither of our updating approaches (gslis_adaptive or gslis_mult) saw much advantage over the other.

2. Our approaches, though significantly stronger than the group-wise mean with respect to F1 operate with effectiveness that is nearly identical to the group-wise median, at least with respect to F1.

Figures 1 and 2 show the performance of several models measured at each month in the KBA corpus. The upper-left panel compares our official runs to a "toy" (i.e. straw man") approach. The toy system simply weights occurrences of the exact entity title at 2.0 and individual words from the title at 1.0, using this weighted feature set as a simple indri query.

A few points of interest emerge from Figures 1 and 2. First, our two methods of updating models largely track in parallel, but do occasionally (e.g. at month 5) give different results. Second, our approaches are only slightly superior to the toy system. But as we can see from the horizontal red line in the figure, the straw man actually performed near the TREC median. Lastly, performance declines over time. Though this appears to be a problem with query drift, we might keep in mind that the toy system's model is in fact static. Instead of suffering problematic query drift, it seems to be that case that our performance declines because the KBA problem becomes more difficult (at least with respect to these metrics) over time. In fact, during conversations with track organizers, it was suggested that new sources of documents entered the corpus in the later months, and that these were in some sense noisier than the initially dominant documents. Thus the performance decline may be an artifact of the data.

Table 4: uiucGSLIS KBA Experimental Results. Statistics are calculated using only "central" documents for relevance .

Run	F1	SU
TREC median	0.289	0.333
TREC mean	0.220	0.311
gslis_adaptive	0.284	0.339
gslis_mult	0.284	0.337

Two points are worth elaborating regarding these data. First, our system was designed explicitly as a ranking system (i.e. a document routing platform). We are thus eager to compare our performance using other teams' results using rank-based measures. Secondly, the success of the "toy" system (c.f. Figure 1) suggests that the community has a good deal of research to do if we hope to perform KBA successfully.

References

[1] M. Efron. Query-specific recency ranking: Survival analysis for improved microblog retrieval. In *SIGIR 2012 Workshop on Time-aware Information Access (#TAIA2012)*, 2012.

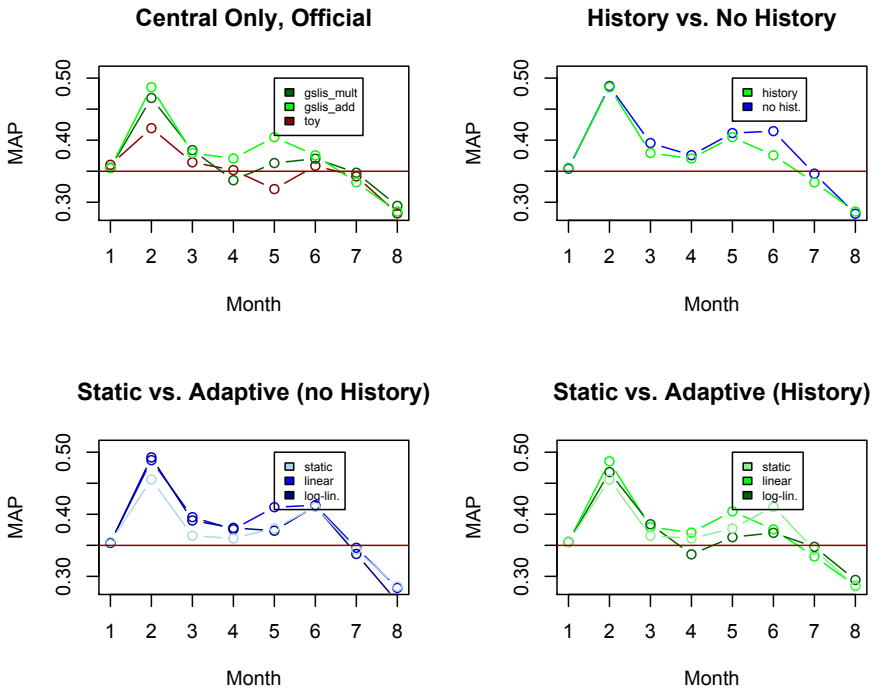

Figure 1: F1 Observed on a Month-by-Month Basis. Red line is a the mean F1 of a "straw man" model.

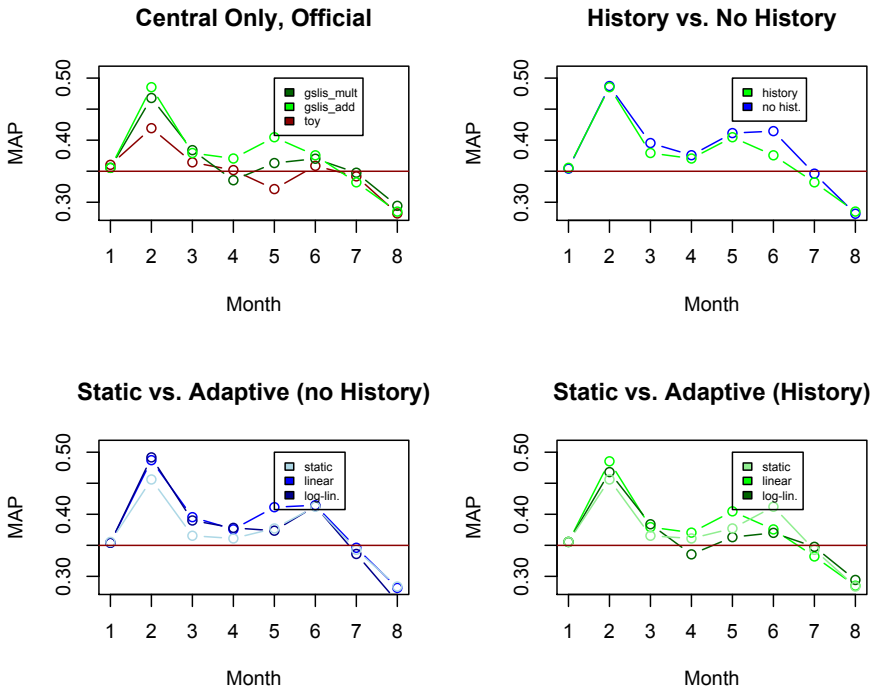

Figure 2: Mean Average Precision Observed on a Month-by-Month Basis. Red line is the mean MAP of a "straw man" model.