

The University of Illinois' Graduate School of Library and Information Science at TREC 2013

Miles Efron, Craig Willis, Peter Organisciak, Brian Balsamo, Ana Lucic
Graduate School of Library and Information Science
University of Illinois, Urbana-Champaign
501 E. Daniel St., Champaign, IL 61820
mefron,willis8,organis2,balsamo1,alucic2@illinois.edu

February 3, 2014

1 Introduction

The University of Illinois' Graduate School of Library and Information Science (uiucGSLIS) participated in TREC's knowledge base acceleration (KBA) track in 2013. Specifically, we submitted runs for the cumulative citation recommendation (CCR) task. CCR is a type of document filtering. The use-case is that our user U is an editor of a node T in a knowledge base such as Wikipedia (for example, the entry for blogger Jamie Parsley (http://en.wikipedia.org/wiki/Jamie_Parsley)). Given an incoming stream of documents \mathcal{D} , the system must emit a binary `route/not-route` decision for each document D_i in real time. In this case, the binary decision signals whether we should recommend D_i to U as a source of information for editing the entity T 's page. In other words, our goal is to monitor \mathcal{D} , signaling to U when we find a document that contains "edit-worthy" information regarding the entity T .

2 Experimental Data

2.1 The 2013 KBA Stream Corpus

The KBA "Stream Corpus" is a collection of timestamped documents published on the web between October 2011 and January 2013¹. Our copy of the data consisted of 11,948 directories, each with one hour of published documents. The stream corpus includes additional metadata and derived data (e.g., part-of-speech tags) that were not used by our system.

¹<http://trec-kba.org/kba-stream-corpus-2013.shtml>

2.2 Target Entities

The CCR task involves monitoring the stream corpus for documents that contain vital information about any of a set of 145 target entities. Each entity is associated with a URL — either a Wikipedia page or Twitter account. This year the entities had additional metadata that was not used by our system.

2.3 Data Reduction

We hypothesized that few vital documents would lack a match or near-match on the surface form name of a given entity. Based on this intuition, we performed a massive culling of the corpus, removing all documents that we judged to be not possibly vital with respect to any of the 145 targets. To accomplish this, we generated one or more Boolean AND statements per entity. Only documents that matched at least one of our Boolean statements were retained for further analysis.

For example, the entity designated by:

```
http://en.wikipedia.org/wiki/Fargo_Air_Museum
```

mapped to the Boolean filter *fargo AND air AND museum*. In creating these filters, any terms with length < 2, as well as all punctuation were omitted. We also omitted Wikipedia qualifiers (i.e., terms in parenthesis) and terms in a list of Wikipedia-specific vocabulary such as *disambiguation*. Entities containing diacritics generated two separate statements. For example:

```
http://en.wikipedia.org/wiki/Gwenaëlle_Aubry
```

yielded the statements *Gwenaëlle AND Aubry* and *Gwenaëlle AND Aubry*.

Documents were filtered based on their “clean visible” text, which was tokenized using a Lucene 4.3 `StandardAnalyzer` with an `EnglishPossessiveFilter`. During processing, we scanned a total of 460,899,683 documents and retained 7,132,550 for indexing.

2.4 Training Data

Track organizers defined a training period as the time from the stream beginning until 1 March, 2013. Teams were permitted to analyze documents from within this “training window,” as well as corresponding ground truth annotations about entity-document relevance.

2.5 Annotation Data

The CCR task in 2013 recognized two levels of relevance with respect to an entity-document pair: *useful* and *vital*. A vital rating indicates a stronger usefulness of the document than a useful rating does. The official goal of the task was to maximize F1 based on vital ratings.

3 Base System

For core indexing and retrieval we used the Indri search engine and API². 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. For a few tasks, stoplists were used; we describe these below. Otherwise, no stopping was performed.

4 Overall Approach

For assessing entity-document similarity, we used the negative KL-divergence between the language model θ_i for document D_i and the θ_E for the entity E :

$$\text{sim}(D_i, E) = - \sum_{f \in E} P(f|\theta_E) \log \frac{P(f|\theta_E)}{P(f|\theta_i)}. \quad (1)$$

where f is a “feature” of the profile we defined to represent E . Usually, some feature f_j is the simply a term that is highly associated with E .

Additionally, $\text{sim}(D_i, E)$ was supplemented with an entity-independent feature ℓ that promotes documents based on their length. Our intuition was that relevant documents will tend to have lengths that are neither very short nor very long. Instead, we suspected that relevant documents would tend to be “well-behaved” with respect to length. We used the training data to find the length of all *vital* or *useful* documents. Let R_T be the set of documents labeled as either *vital* or *useful* in the training data. If we let $L(D_i)$ be the length of document D_i , we hypothesized that $L(D) \in R_T$ would follow a log-normal distribution \mathcal{N}_L . Using R_T , we estimated the mean and standard deviation of \mathcal{N}_L by maximum likelihood, such that

$$\ell(D_i) = \mathcal{N}_L(\log n(D_i), \hat{\mu}, \hat{\sigma}) \quad (2)$$

where $n(D_i)$ is the length of D_i , and $\hat{\mu}, \hat{\sigma}$ are the estimated parameters (mean and standard deviation) of the log-normal distribution.

This yields our final score for D_i against E :

$$\text{score}(D_i, E) = \text{sim}(D_i, E) + \log \ell(D_i). \quad (3)$$

Conceptually, in Eq. 3, $\ell(D_i)$ plays the role of a prior over documents. However, use of KL divergence for measuring similarity does not lend itself to the proper introduction of a prior (unlike, say, query likelihood).

The decision to emit `true` for D_i hinged on the magnitude of $\text{score}(D_i, E)$ with respect to a threshold τ . We emit `true` iff $\text{score}(D_i, E) \geq \tau$.

²<http://lemurproject.org>

Unless otherwise noted, τ was set simply by scoring all training documents according to Eq. 3 and finding the cutoff that maximized the F1 score with respect to the training annotations. When finding the optimal cutoff, some of our runs (those with the word *vital* in the name) calculated F1 using only vital documents as positives, while the others used both vital and useful in finding τ .

5 Entity Representation

Our main research goal this year was to improve CCR effectiveness by building high-quality representations of each target entity. Entity representations were constructed by obtaining an initial representation and then (optionally) changing it in light of various sources of evidence.

To make this discussion clearer, we introduce some notation here. Let θ_i be the representation of entity E_i . In our case θ_i is a language model over the vocabulary of the corpus. Also, let T_i be a sample of text that we obtain automatically at the start of training. The text in T_i is the data that we use to learn our initial estimate $\hat{\theta}_i$. We used different sources to obtain T_i depending on the entity type:

- Wikipedia entities: Text was acquired by cleaning the HTML of the last version of the Wikipedia entry for E_i available before the beginning of the training period.
- Twitter entities: The Twitter API was used to obtain the `name`, `screen_name` and `description` field of the entity’s Twitter profile³.

Based on T_i , we estimated our initial language model by simple maximum likelihood. Thus we have:

$$\hat{\theta}_i^{ml} = \frac{n(f, T_i)}{n(T_i)} \quad (4)$$

where $n(f, T_i)$ is the number of times feature f appears in the training data and $n(T_i)$ is the number of tokens in the training data.

When building profiles, we applied a stoplist that consisted of the standard `indri` stopwords plus 18 additional words that were common in Web documents (e.g. `http`, `www`, `px`).

5.1 Improving Models with Training Judgments

To improve our entity profiles, we created models that combined the initial profile $\hat{\theta}_i^{ml}$ with a second model $\hat{\theta}_i^R$ which was obtained by analyzing the training documents marked as

³Admittedly, these data constitute “future” evidence in the sense that we harvested them after the timespan of the corpus. However, the Twitter profiles that we harvested were extremely terse, and we believe that their inclusion did not give any future-specific advantage to our runs.

vital in the training annotations. This gives us:

$$\hat{\theta}_i^R = \frac{n(f, R)}{n(R)} \quad (5)$$

where $n(f, R)$ is the frequency of feature f in the relevant documents R , and $n(R)$ is the total number of tokens in all documents comprising R . In other words, our “feedback model” is simply the maximum likelihood estimator of the multinomial obtained by considering all relevant training documents as one long pseudo-document. Before using our estimated feedback models, we truncated them, retaining only the $k = 50$ most probable words in them.

In the runs that we called “feedback runs,” we linearly interpolate our two models via:

$$\hat{\theta}_{iF} = \lambda \hat{\theta}_i^{ml} + (1 - \lambda) \hat{\theta}_i^R \quad (6)$$

where we simply set $\lambda = 0.5$.

5.2 Dynamic Model Updating

One of our chief research questions was: can we improve the expressiveness of entity models *after* the training period? In other words, we hoped to use our experience during the testing phase to update our models. Of course this was difficult since any post-training learning is necessarily unsupervised. Our motivation for allowing models to “evolve” stemmed from two factors. First, for many entities the vital documents in the training corpus were rare or non-existent. Without adequate training data, any learned profile is likely to be over-fitted and weak. Since it is likely that we will see more relevant data during the evaluation time period, we stand to improve the initially poor models if we can marshal these data.

Our second motivation lays in the dynamic nature of the KBA task itself. Over the course of the KBA task, emerging events may cause new terms to become associated with vital information related to an entity. We reasoned that a model should be able to adapt in a way that promotes the *current* distribution of terms used to discuss an entity.

To support evolving models, we let the profile obtained at the end of the training period, θ_i (here, θ_i is shorthand; it might be $\hat{\theta}_i^{ml}$, $\hat{\theta}_i^R$, etc.) induce a Dirichlet prior over terms, along with the concentration parameter μ . Then, terms found in *predicted* vital documents were used to update the model. Thus at time t during the evaluation period, let \mathbf{V}_t be the set of predicted vital documents we have accumulated for entity E_i so far. In our dynamic runs, we then have the model:

$$\hat{\theta}_{it} = \frac{n(f, \mathbf{V}_t) + \mu P(f|\theta_i)}{n(\mathbf{V}_t) + \mu} \quad (7)$$

for some concentration hyperparameter μ , whose magnitude governs the extent to which we allow the model to diverge from the initial model θ_i .

One additional notion that we implemented is based on the observation that the size of \mathbf{V}_i can never decrease. As it increases, the predicted vital documents will overwhelm the influence of the initial—putatively good—model. In a Bayesian sense, this is the correct behavior—as we learn more about E_i , we rely more on our data and less on our prior. However, this intuition is problematic because of the unsupervised nature of our updating. Without recourse to any ground truth, it is likely that \mathbf{V}_t will contain some non-vital documents, despite our best efforts. Thus, updating is risky insofar as we are actually sampling documents from two populations—vital and everything-else—while we only wish to sample from the vital population.

To mitigate this risk, we implemented a heuristic that the updating process is limited by a constrained “memory” window. That is, we choose a number of documents w , and we limit \mathbf{V} to contain no more than w of the most recently emitted documents. Once \mathbf{V} contains w documents, when we emit a new document, the oldest member of \mathbf{V} is discarded and replaced with the new document.

5.2.1 Overview of Model Updating

Here we briefly describe how we approached CCR using dynamic models:

1. Use the training data and possibly training judgments to learn our initial model θ_0 .
2. Train our initial cutoff threshold τ_0 using θ_0 and the training data.
3. Begin iterating over test documents.
4. When, at time t , we find a document D with a score greater than τ_0 we emit D and then enter the updating procedure:
 - (a) Add D to \mathbf{V} .
 - (b) Estimate $\hat{\theta}_{it}$.
 - (c) Using the newly estimated $\hat{\theta}_{it}$, generate a new threshold τ_t over the training documents.
 - (d) Remove any old documents from \mathbf{V} if needed.
 - (e) Return to iteration over evaluation corpus.

6 Submitted Runs

This year we submitted 11 runs. Results of these runs are shown in Table 1. The names of the runs shown in the table are intended to be self-explanatory after introducing some basic vocabulary. First, we submitted two broad classes of runs:

- *static*. Runs where the entity model at the end of the training period was fixed and did not change during the evaluation period.

- *dynamic*. Runs where the entity model was allowed to adapt over the course of the evaluation period via the Bayesian updating method described above.

Unless otherwise specified, all runs began with a profile estimated using the relevance feedback method described above. The exception was `static_vital_wikihtml`. In this case, models were simply normalized word counts from Wikipedia (or Twitter) entries.

All runs labeled *bayes_** were adaptive. They varied along two dimensions:

- *Window size*. During updating, only most recent k emitted documents were stored for training the models. Window size refers to k . Larger values of k indicate a longer “memory.”
- *Prior Strength*. The Dirichlet concentration parameter μ governed the extent to which new information changed our models. Larger μ values gave more weight to the prior (the original model), constraining change over time.

The only other run type was `static_fb_vital_smarthThresh`. This was a heuristically implemented run that attempted to compensate for cases where the threshold obtained by optimizing F1 over the training data was sub-optimal. In this case, given that vital documents *were found* for an entity during training, the run was identical to `static_fb_vital`. But in those cases where we found no vital entities, all other runs would set $\tau = -\infty$. To improve on this, the `static_fb_vital_smarthThresh` replaced all cases where $\tau = -\infty$ with the 95th quantile of (non-vital) scores seen during the training phase.

7 Analysis of Results

Tables 1 and 2 summarize the effectiveness of our runs.

Little in the way of systematic differences are obvious from these data. For instance, neither static nor dynamic models show an across-the-board advantage. Also, the best-performing run for the *vital only* condition scored near the bottom using the *vital+useful* rubric.

Our chief interest is the comparison of the effectiveness of the static v. dynamic models. The various instantiations of each of these two types was less important. To help identify any intrinsic difference between static and dynamic models, Figure 1 shows results from three static runs (top panels) and three dynamic runs (bottom panels). Each bar in the bar plots corresponds to one KBA entity. And the bar height is the number of vital documents enumerated in the ground truth annotations minus the number of predicted vital documents by a given run. Thus, a bar height of 0 means that the run emitted precisely the same number of vitals as there actually were. The dotted red lines in the plots are reference points of ± 100 . The aim of this figure is to show any systematic tendency to over- or under-emit.

Table 1: Results of Official GSLIS KBA CCR Runs. Vital Only. Runs are shown in decreasing order of F1.

Run Name	Precision	Recall	F1	SU
static_fb_vital_tfidf	0.2370	0.2999	0.2648	0.2630
static_fb_vital_smarthThresh	0.2370	0.2999	0.2648	0.2630
bayes_window.20_mu.10000	0.2382	0.2758	0.2556	0.2774
static_vital_wikihtml	0.1855	0.3953	0.2525	0.2577
bayes_window.200_mu.2500	0.2291	0.2780	0.2512	0.2701
bayes_window.20_mu.2500	0.2269	0.2804	0.2508	0.2678
static_fb_vital	0.2083	0.3140	0.2504	0.2752
bayes_window.50_mu.1500	0.2083	0.3140	0.2504	0.2752
bayes_window.20_mu.5000	0.2083	0.3140	0.2504	0.2752
bayes_window.20_mu.1500	0.2322	0.2694	0.2494	0.2738
bayes_window.20_mu.150	0.2432	0.2546	0.2488	0.2729
bayes_window.50_mu.2500	0.2243	0.2783	0.2484	0.2658

Table 2: Results of Official GSLIS KBA CCR Runs. Vital+Useful. Runs are shown in decreasing order of F1.

Run Name	Precision	Recall	F1	SU
static_vital_wikihtml	0.5319	0.4187	0.4686	0.4882
static_fb_vital	0.5540	0.3208	0.4063	0.4610
bayes_window.50_mu.1500	0.5540	0.3208	0.4063	0.4610
bayes_window.20_mu.5000	0.5540	0.3208	0.4063	0.4610
bayes_window.200_mu.2500	0.5763	0.2878	0.3838	0.4434
bayes_window.50_mu.2500	0.5724	0.2885	0.3836	0.4414
bayes_window.20_mu.2500	0.5735	0.2869	0.3824	0.4443
static_fb_vital_tfidf	0.5329	0.2902	0.3758	0.4444
static_fb_vital_smarthThresh	0.5329	0.2902	0.3758	0.4444
bayes_window.20_mu.10000	0.5836	0.2752	0.3740	0.4414
bayes_window.20_mu.1500	0.5726	0.2765	0.3729	0.4424
bayes_window.20_mu.150	0.5843	0.2252	0.3251	0.4230

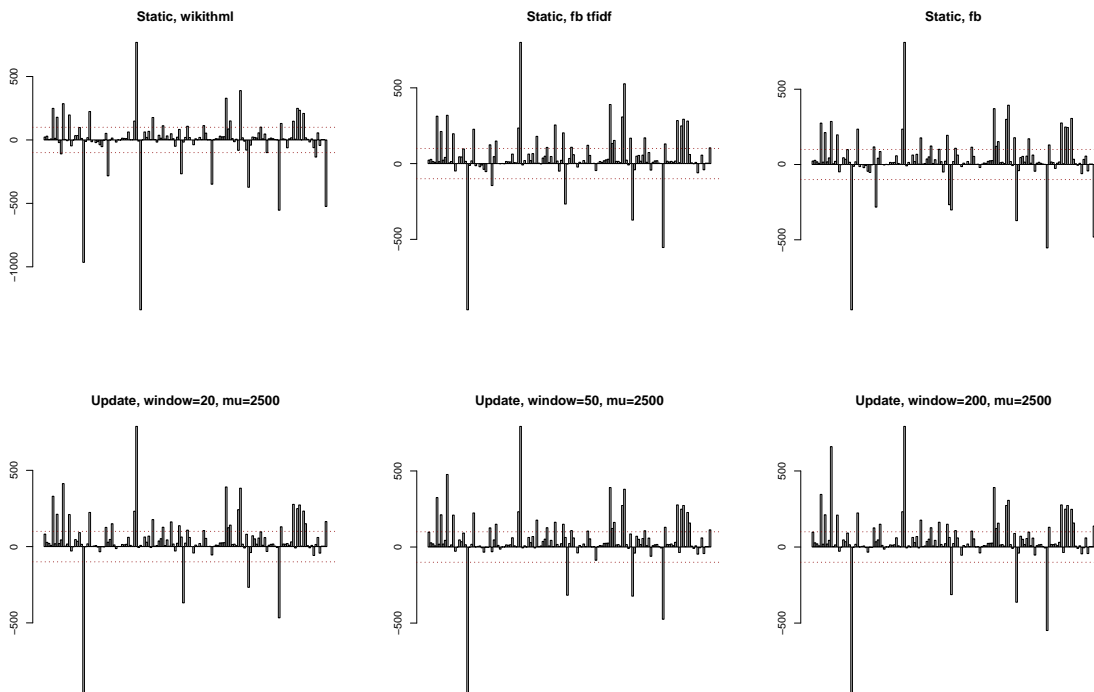


Figure 1: Number of Vital Documents per Entity (Existent and Emitted). Each panel corresponds to one uiucGSLIS CCR run. Within panels, each bar is one KBA entity. Bar height is the true number of vital documents per entity minus the number of predicted vital per run.

The figure shows very little difference among runs. The only clear exception is the case of the `static_vital_wikihtml` run. While all runs show one entity with an outsized false positive rate, `static_vital_wikihtml` shows two such entities.

Also, the dynamic models had fewer large false positives than the static models. For instance `bayes_window.20_mu.2500` only over-estimated the number of vitals by more than 100 four times, while the static approaches found far too many false vitals seven or eight times.

We were surprised to see a negligible difference between the static and dynamic modeling approaches. Figure 7 suggests one mechanism that may have diluted any observable differences. The figure shows changes in two entity models—`CorbinSpeedway` and `Jennifer_Baumgardner`—over the span of the evaluation period. The figure’s left panels (in black) give the KL divergence between the model at time x and θ_i , the initial model estimated from the training documents. The right panels (in red) show the KL divergence

between the model at time x and the background language model (of the training corpus).

Increases in the black graphs indicate that the model is moving “away” from the initial model. However, they say nothing about whether this motion is for the good or the bad. In red, decreases indicate that the model is becoming more similar to the background language model. In more familiar terms, as the red graphs decrease, we are losing “query clarity.” Thus, increased divergence from θ_i (black graphs) is possibly helpful or hurtful, decreased divergence from the background model is probably harmful, indicating drift or “dilution” of the model.

Though the dynamics vary between the two entities, in both cases, the end result is the same: the model moves away from the initial profile, winding up much closer to the background model than it started. This suggests that over time, the models are drifting off course.

However, the plots referring to the Jennifer_Baumgardner models are interesting. They suggest that the updating process is doing more than (or at least something besides) simply drifting towards the background model. Instead, we see a strong spike in divergence from *both* the initial model and the background model early on. This is followed by a rapid return to starting-model similarity and a much slower return to drifting toward the background.

8 Conclusion and Future Directions

Our next efforts will include two main tasks. First, we will undertake more intensive data analysis. We have only presented a cursory overview in this notebook paper. Understanding what’s happening in Figure 1 will require more analysis.

Additionally, we plan to make improvements in the model updating procedure described here. While probabilistically simple, the learning method that we used was clearly not optimal. As we might expect, adding pseudo-relevant documents on a wholesale basis into our model seems to have introduced more noise than we can tolerate. In future work we plan to further constrain the updating process.

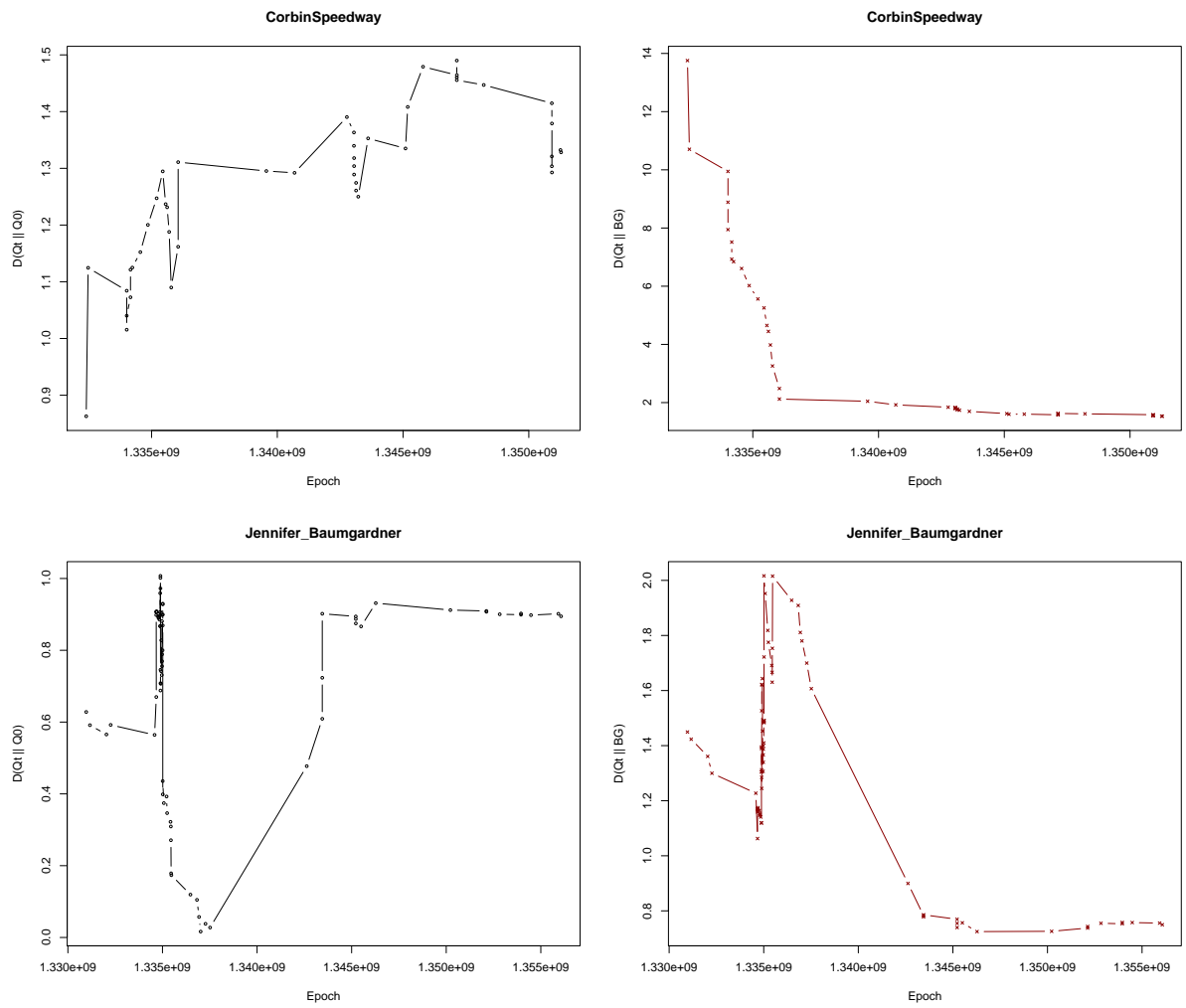


Figure 2: Entity Model Clarity and Resilience. The top row shows results for the Corbin-Speedway entity, with Jennifer_Baumgardner in the bottom row. Left-hand columns give the KL divergence between the entity profile at time t (the x -axis) and the profile from the start of the ETR. Right panels given the KL divergence between the entity profile and the language model of the training data's full collection.