Large-Scale High-Precision Topic Modeling on Twitter

Shuang Yang
Twitter, Inc.
syang@twitter.com

Alek Kolcz
Twitter, Inc.
ark@twitter.com

Andy Schlaikjer
Twitter, Inc.
hazen@twitter.com

Pankaj Gupta
Twitter, Inc.
pankaj@twitter.com

ABSTRACT

We are interested in organizing a continuous stream of sparse and noisy texts, known as "tweets", in real time into an ontology of hundreds of topics with *measurable* and *stringently high* precision. This inference is performed over a full-scale stream of Twitter data, whose statistical distribution evolves rapidly over time. The implementation in an industrial setting with the potential of affecting and being visible to real users made it necessary to overcome a host of practical challenges. We present a spectrum of topic modeling techniques that contribute to a deployed system. These include non-topical tweet detection, automatic labeled data acquisition, evaluation with human computation, diagnostic and corrective learning and, most importantly, high-precision topic inference. The latter represents a novel two-stage training algorithm for tweet text classification and a close-loop inference mechanism for combining texts with additional sources of information. The resulting system achieves 93% precision at substantial overall coverage.

Categories and Subject Descriptors
I.2.6 [Artificial Intelligence]: *Learning*

1. INTRODUCTION

Twitter [1] is a global, public, distributed and real-time social and information network in which users post short messages called "*tweets*". Users on Twitter *follow* other users to form a network such that a user receives all the tweets posted by the users he follows. Tweets are restricted to contain no more than 140 characters of text, including any links. This constraint fosters immense creativity leading to many diverse types of styles and information carried in the tweets. As of early 2014, Twitter has more than 240 million monthly active users all over the world. These users send more than 500 million tweets every day, which corresponds to an average of 5700 tweets per second, with spikes at up to 25 times of that velocity during special events [2].

Tweets generated from all over the world are expected to be about a variety of topics. Figuring out exactly which tweet is about which topic(s) is interesting to us at Twitter in our goal to serve our users better as it enables personalization, discovery, targeted recommendations, organization of content, as well as aiding in studies and analyses to gain better insights into our platform as a whole. Such broad usefulness, and the potential of affecting and being visible to real users, however, propose demanding precision requirement. In this paper, we consider the problem of high-precision topic modeling of tweets in real-time as they are flowing through the network. This task raises a set of unique challenges given Twitter's scale and the short, noisy and ambiguous nature of tweets. We present how we address these challenges via a unique collection of techniques that we have employed in order to build a real-time high-precision tweet topic modeling system that is currently deployed in production inside Twitter. Our system achieves 93% precision on $\sim$300 topics and 37% overall coverage on English tweets.

2. OVERVIEW & RELATED WORK

Given a tweet, and all the information associated with it (e.g., author, entities, URLs, engagements, context), we are interested in figuring out what topic(s) this tweet is about.

Topic modeling tasks have been commonly approached with unsupervised clustering algorithms (e.g., k-means, pLSI and LDA [5]), information filtering approaches (e.g., language models [19]), or weakly supervised models (e.g., supervised LDA [5], labeled LDA [22]). These approaches are effective in grouping documents into a predefined number of *coarse clusters* based on inter-document similarity or the co-occurrence patterns of terms (with help from very mild supervision). They are also cheap to train as no or very few labeled data is required. Nevertheless, they are not suitable for our use because it is very difficult to align the topic clusters produced by these approaches to a predefined ontology and perform low-latency topic inference with measurable and decent precision. In fact, *none* of these approaches could attain the precision close to what we demand.

Instead, we primarily consider supervised approaches that classifying tweets into our taxonomy with controllable high-precision. Figure 1 illustrates the overview of our system. On the training path, our "training data collector" constantly listens to the tweet stream to automatically acquire labeled training data, which, once accumulated to a certain amount, are fed into the "trainer" module to produce classification models; These models are then validated and calibrated to be used in our service. On the operation path, the classifier

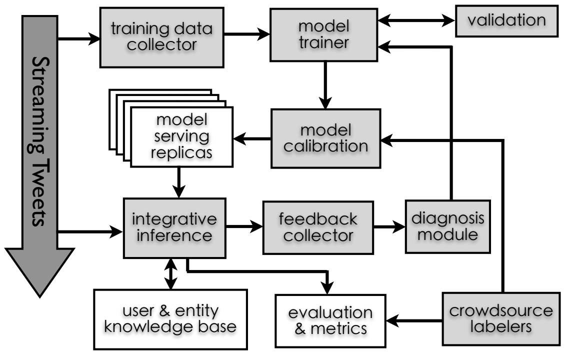


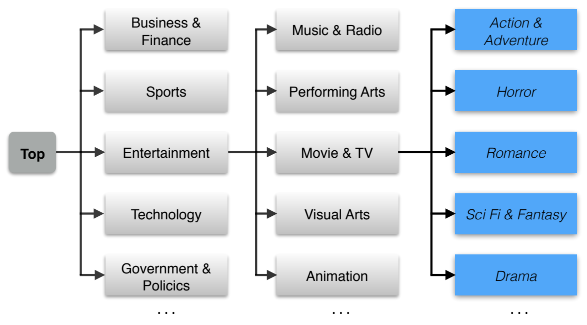
Figure 1: Overview of tweet topic inference system.

models are replicated and served in parallel, together with knowledge about users and entities, in a "integrative inference" module to provide high-precision topic annotation of tweets in real time. The inference results are used to generate and refine the user-and-entity knowledge, making this a close-loop inference. The results are also scribed and used to evaluate and monitor the quality performance with help from crowd-sourced human annotation. Once the service is exposed to the user, we also employ a "feedback collector" and a "diagnosis module" to gather high-precision user curated data, which are fed back into the trainer module to fine-tune the trained models.

Related Works & Contributions. Topic modeling of tweets has been examined extensively in the literature. Most existing research has been based on un- or weekly supervised techniques such as variations of LDA [17, 27]. The inference in these approaches usually takes considerable latency, and the results are not directly controllable when precision is concerned. Recent studies employ information filtering approaches to detect tweets on specific topics of interests[19, 9]. These approaches usually have low latency and can perform inference in real-time; and when configured properly, the top-ranked results obtained by them can usually achieve a reasonable precision target. Nonetheless, because they identify only a small fraction of tweets that are *apparently relevant* to a given topic while disregarding the majority rest, the recall and coverage of these approaches are very poor, yielding heavily biased inference results that are less useful in many of our use cases. Information filtering approaches are also *only* good at specific topics, usually topics of relatively focused meanings (e.g., "San Francisco" and "NBA"), and *not* directly applicable to a broad spectrum of topics such as an ontology, as what we study here. To the best of our knowledge, this is the first published documentation of a deployed topic modeling system that infers topics of short noisy texts at high precision in real-time.

3. TAXONOMY CONSTRUCTION

We create a taxonomy represented as a fixed (and potentially deep) ontology to encode the structure and knowledge about the topics of our interest. The goal is to find a topic structure that is as complete as possible to cover as many semantic concepts (and their relationships) as we can, while focusing on topics that are frequently discussed on Twitter



Similar to Assembly AI's list

Figure 2: A subtree of our taxonomy. The deepest node is 6 hops way from the root node "Top".

and are less likely to be transient. Starting from existing taxonomies such as ODP and Freebase, we undertook a number of exploratory approaches and manual curations. For example, by building topic filters based on ontology corpora (e.g., ODP, Wikipedia), we can estimate coverage of topics on Twitter and trim or merge topics that are too niche; unsupervised models such as LDA were also used to map clusters to well defined ontology (e.g., ODP, Freebase) in order to estimate coverage. This process has been iterated through multiple refinements and over different time periods to avoid bias. The current taxonomy consists of ~300 topics with a maximum of 6 levels, a fragment of which is shown in Figure 2.

300 topics ; 6 levels

4. TWEET TOPIC MODELING: TEXT CLASSIFICATION

A tweet can be associated with multiple modalities of information such as text, named entities, users (the author and people who engaged with the tweet), etc. In this section, we consider topic classification of tweets based solely on its textual features, which will be combined with other sources of information for integrative topic inference in Section 5.

High-precision tweet classification faces unique challenges because unlike ordinary documents, tweets are much sparser and noisier and the scale is daunting. The demanding precision criteria make this even more challenging. A tweet may belong to multiple topical categories at the same time, or be non-topical, which makes this a multi-label classification problem. To meet quality criteria, the participating classifier needs to be abstaining, i.e., providing topic labels only if the expected precision or confidence is sufficiently high.

4.1 Chatter detection

One key challenge for high-precision tweet classification is that a substantial proportion of tweets are non-topical [3], or at least their topics are not clear in the context of a single tweet. We use the term *chatter* to denote tweets that often primarily carry emotions, feelings or are otherwise related to personal status updates. While they are clearly an important part of user expression on Twitter, it is necessary to reject classifying these tweets (e.g., to save system latency and to control quality) and restrict the domain of interest to tweets that are *not* chatter. To that end, we build content filters to eliminate chatter tweets, as described in previous work [3].

4.2 Training data acquisition

Training high-quality tweet classification requires high-quality labeled data. Use of human annotation is prohibitive for a number of reasons. First, it is too expensive: for the scale we consider (i.e., 300 topics, millions of features) and the sparseness (i.e., only ∼10s feature presence per tweet), any reliable estimation would require hundreds of millions of labeled tweets and cost millions of dollars even with the cheapest crowd-sourced service in the market. Moreover, even human-assigned labels could be too noisy. A big taxonomy like ours presents considerable *cognitive overload* to human annotators, for example, when asked to assign relevant topic labels (out of the 300 candidates) to each tweet, human raters turn to identify only a fraction of the true positive labels. While it is possible to obtain samples of data within each category, such samples are likely to be biased and to the extent that some of these instances also belong to other categories, most of these memberships will remain unknown. Human labeled data are also subject to expertise bias because a lot of niche topics (e.g., "data mining") exceed the literacy of crowd-source workers, forcing them to respond with random-guess annotations. Because of these concerns, in our system, human labeled data (with enhanced quality assurance) are only used in quality evaluation (Section 4.7). For model training, we devise scalable algorithms to collect labeled data automatically from unlabeled tweets.

The labeled data acquisition pipeline is illustrated as Figure 3. First, tweets are streamed through a set of *topic priors* to prefilter tweets that are weakly relevant to each topic. These topic priors are white-list rules that include:

User-level priors: users who tweet predominantly about a single topic (e.g., `@ESPN` about "Sports").

Entity-level priors: unambiguous named entities and hashtags that are highly concentrated to one or a few topics (e.g., `#NBA` about "Basketball"), see Section 5.2.

URL-level priors: for tweets containing URLs, the URL string usually contains topic information about the webpage it links to (e.g., `sports.yahoo.com/nba/*.html`).

We also leverage the **social annotation** functionality to extract topic priors. Many Twitter users represent authorities on a small set of topics by virtue of their profession, experience, and the fact that they tend to publish on those topics on/off Twittter. This is recognized by other Twitter users who sometimes group users "known for" a particular topic using Twitter's List feature.

The data coming through the topic prior filters are only weakly relevant to each topic and thus very noisy. To obtain high-quality positive data for each topic, we employ a *co-training* [6] based data cleaning algorithm. In particular, we consider only these tweets containing URLs, and iteratively apply the following for each topic c:

1. train a webpage classifier to remove tweets whose associated URLs satisfy: $p(c|\text{URL})$ is below ϵ_1-percentile.
2. train a tweet classifier to remove URLs whose associated tweets satisfy: with $p(c|\text{tweet})$ is below ϵ_2-percentile.

☆ The key assumption here is that when a URL is embedded in a tweet, the webpage that the URL links to is highly likely to be on the same topics as the tweet. To make this procedure more scalable and less coupled with the final classifier models, we use Naive Bayes classifiers in this step.

→ Assumption for cleaning.

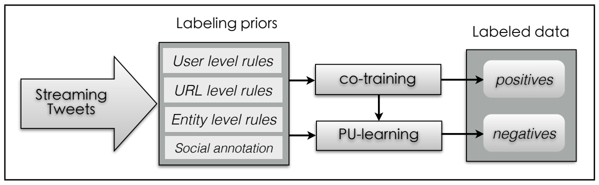


Figure 3: Training data acquisition pipeline.

Once we have positive data, for most topics, negative examples are very easy to collect, e.g., by using large-amount of randomly sampled tweets or using "*one-vs-all*" splitting of the positive data. However, for a considerable number of relatively broad topics that receive high velocity of coverage on Twitter (e.g.,"Sports", "Technology" and "News"), randomly sampled negatives are very noisy. One-vs-all doesn't work well for topics that are intercorrelated (e.g., "Sports" and "Basketball"), because semantically equivalent documents can be presented to a learner as both positive and negative examples. To this end, we employ algorithms developed for *learning from positive and unlabeled data* (or PU-learning [12]) to acquire high-precision negative examples for each topic. In particular, we select tweets / webpages as negative instances for a particular topic only when the similarity scores with the centroid of that class are below (1-ϵ_3)-percentile, i.e., via a *Rocchio* classifier [20].

4.3 Feature extraction

The effectiveness and efficiency in feature extraction are vitally important to the success of our system given our scale and quality requirement. Conventional unigram text features cannot meet our needs in both regards. First, tweets are short (bounded with 140 characters), and amount to merely ∼7 unigram terms (including common stop-words) on average – very difficult for any machine learning model to infer topics with high confidence. Second, traditional implementation of unigram extraction can take considerable computational resources and dominate our system latency especially for webpages which could exceed 10K terms, which is problematic as we require real-time scoring. The latter factor is actually more critical and essentially prohibits us from using potentially more effective features such as n-grams, *word2vec* [21], or other time-consuming feature processing such as *term pre-selection* and *stemming*. Instead, in our system, we use the following two types of feature extractions for tweets and webpages respectively:

Binary hashed byte 4gram We use a circular extractor that scans through the tweet text string[1] with a sliding window of size four: every four consecutive bytes (i.e., UTF8 chars) are extracted and the binary value (4-byte integer) is hashed into the range of 0 to $d-1$, where d is chosen to be a prime number to minimize collision (e.g., $d = 1,000,081$ in our experiment). The occurrence of these hashed indices are used as binary features in tweet classifiers. This extractor (`Byte4Gram`) yields significantly denser feature vectors than unigram, e.g., for a tweet of length l, it produces exactly l feature occurrence.

[1]Tweet texts are preprocessed at the time of scoring such that less meaningful strings such as URLs and `@mentions` are stripped off and all strings are converted to lower case.

Hashed unigram frequency Term unigrams are extracted and hashed to integers at an extremely fast speed [13], with also a feature size d of $\sim$1 million. The term frequencies of the indices are then transformed to log-scale, i.e., $\log(1 + \text{tf}_w)$ and the feature vectors are normalized so that each webpage has the same ℓ_1-norm. This extractor (`Unigram-logTF-norm`) is extremely fast and it doesn't require any string preprocessing as Unicode conversion, lowercasing and word boundary detection are automatically taken care of.

We found that these feature extractors achieve the best balance of effectiveness and efficiency in our system.

4.4 Model pretraining

Both tweet and webpage topic inference can be casted naturally as multi-class multi-label classification problems, making *regularized logistic regression* a perfect fit. Given a set of n training instances $x_i \in \mathbb{R}^d$ together with their corresponding labels $y_i \subset \{1, 2, ..., C\}$, where C is the total number of topics, we consider two types of LR models:

Multinomial logistic regression (MLR) also known as *Multinomial logit* or *Softmax Regression* model, which models the conditional topic distribution given an instance x as the following multinomial distribution:

$$p(c \in y|x) = \frac{\exp(w_c^\top x + b_c)}{\sum_{c'=1}^{C} \exp(w_{c'}^\top x + b_{c'})}, \forall c = 1, \ldots, C.$$

One-vs-all logistic regression (LR) models "whether x belongs to a topic class c" as a Bernoulli variable:

$$p(c \in y|x) = \frac{1}{1 + \exp(-w_c^\top x - b_c)}, \forall c = 1, \ldots, C.$$

Let $l(w, b|x, y)$ be the log-likelihood of parameter (w, b) given example (x, y). To estimate the parameters (w, b), we minimize the negative log likelihood plus a penalty term:

$$\min_{w,b} \; \lambda(\alpha||w||_1 + \frac{1-\alpha}{2}||w||_2^2) - \frac{1}{n}\sum_{i=1}^{n} l(w, b|x_i, y_i), \qquad (1)$$

where the parameter λ controls the strength of regularization. Here we consider the ElasticNet regularization which is a hybrid of ℓ_1 and ℓ_2 regularization types and includes both as special cases (when α takes 1 or 0). The non-differentiable nature of this regularization, when $\alpha > 0$, enforces sparsity of the model w, which is valuable to us because (1) a compacter model consumes less memory, loads faster and has better latency in real-time scoring; and (2) less useful (e.g., redundant or noisy) features will be automatically ruled out in training, which is important to us as we do *not* do attribute prepruning or stemming.

In our machine learning library, we provide a number of versatile model trainers on Hadoop, including distributed streaming training via SGD [18], and batch-mode training based on *L-BFGS*, *conjugate gradient* (CG) or *coordinate descent* (CD) algorithms. We also provide full regularization path sweeping for ℓ_1 and ElasticNet regularization.

I should read about these

The major difference between the two LR models lies in the Softmax function MLR employs to normalize the posterior scores $p(y|x)$. There are both pros and cons for doing so. On the one hand, MLR provides comparable topic scores which enables us to apply *max-a-posterior* inference to increase topic coverage. As topics are competing against one

Table 1: Streaming training (e.g., Pegasos) generates models with lower AUCs than batch-mode training. Increasing the number of iterations (i.e., number of scans over the data) slowly improves the AUCs but makes the training time much longer.

	#iteration	Avg AUC	Training time
Batch	NA	baseline	baseline
Streaming	1	-1.9%	-62%
Streaming	3	-1.2%	+57%
Streaming	10	-1.0%	+663%

Table 2: Comparison of MLR and LR 1-vs-all classifiers, in terms of average AUC across topics and the % of topics with improved AUCs (%topic win).

	Model	Avg AUC	%topic win
Tweet	MLR	baseline	baseline
Tweet	LR	+3.9%	86%
Webpage	MLR	baseline%	baseline
Webpage	LR	+1.7%	77%

another, the inference results of MLR are less ambiguous (e.g., if topic c fires for a tweet x, topic $c' \neq c$ is less likely to fire for x due to the "*explaining-away*" effect). On the other hand, however, MLR requires the label space to be exhaustive (i.e., a tweet must belong to at least one of the topics), and it discourages usage of examples with missing label, which is not perfectly compatible to our setting. Also, because topics are coupled with one another in MLR, it is hard to train models for multiple topics in parallel, or retrain model for a subset of topics without affecting others.

Experiment Results. We use AUC (i.e., *area under the ROC curve*) for model validation for its independence of the choice of decision threshold – evaluation of calibrated models at their desired thresholds are reported in Section 4.9. We test models on a held-out subset of the "training data acquisition" output data, and estimate AUC with confidence intervals using standard *Bootstrap*. Because the standard errors are very small, we report here only the AUC scores.

Historically, our classifiers are trained in streaming-mode on Hadoop [18] via stochastic optimization, such as Pegasos [26]. Streaming training consumes a constant amount of memory regardless of the scale of the training data, and it is very fast when the data is scanned through only once. However, we noticed that the model generated by streaming training are significantly worse than batch-mode trained ones (e.g., via L-BFGS, CD or CG). For example, on tweet classification (with `Byte4Gram` feature), Pegasos-trained models, with 1 scan over the data, are 1.9% worse on average than batch-mode trained model, as seen in Table 1. Increasing the number of iterations helps to make the gap smaller, but at the same time, also makes the training time much longer (e.g., overhead in data I/O and communication). As we note in Section 4.10, we were able to scale up batch-mode training on massive data with distributed optimization. Hence, in the following experiments, the models are all trained in batch-mode unless noted explicitly.

Besides the advantages we mentioned previously, we found that 1-vs-all LR also performs better than MLR on both tweet and webpage classification. The results are summa-

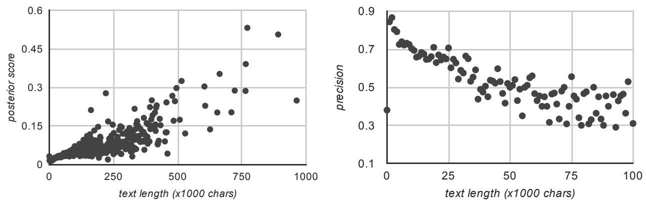


Figure 4: Effect of text length on LR classifier posterior score and precision of webpage classification. Left: longer texts receive higher posterior scores regardless of relevance; Right: precision drops as texts get longer when no feature normalization is used.

rized in Table 2, where all the models use `Byte4Gram` features. On tweets, using LR improves the AUC on 86% of the topics and by 3.9% on average across all topics. Interestingly, the improvements on webpage is relatively small, i.e., only 1.7%. This is due to LR's sensitivity to the variations in document lengths. Recall that here we use `Byte4Gram` features without normalization. When this is used in MLR, the softmax function automatically normalizes the posterior scores. However, in LR, if a text is getting longer, it gets more feature occurrences and in turn receives higher posterior scores regardless of the relevance of the documents. This effect is negligible on tweets as the lengths are bounded, but it is very dramatic on webpage and significantly deteriorates the prediction quality (e.g., precision) on lengthier webpages, as shown in Figure 4. Indeed, when we added feature normalization to webpage classifier, we observed up to 1.5% more improvement on AUC, bringing the overall improvement to 3.2%. The comparative results on feature extractors[2] are summarized in Table 3. Because of their superior performance, we use `HashedByte4Gram` for tweets and `Unigram-logTF-norm` for webpages as default features.

More feature occurrences in longer text

This helped

Finally, using ElasticNet with regularization path sweeping (i.e., optimized λ), further improves average AUC by 0.75–0.98%, as shown in Table 4. Due to the sparse nature of tweets, we found that relatively dense models have slightly better AUCs – models that are too sparse tend to generalize poorly on `Byte4Grams` that are not seen in training data. In our experiments, we found $\alpha \approx 0.05$ provides the best trade-off between test set AUC and training speed (smaller α turns to slow down the regularization sweeping).

4.5 Relational regularization

The ontological relations among topics, as defined by the hierarchical tree structure of our taxonomy, can be used to make our models smarter in learning classifiers for conceptually related topics. We consider three approaches here:

Label expansion This approach regularizes the learner by applying ontology-based label propagation to the training data, i.e., via: (1) *ancestor inclusion*: if x belongs to topic c, then it should also belong to topic c' if c' is an ancestor of c; and (2) *offspring exclusion*: if x belongs to c, then it should *not* be used as a negative example of c' in 1-vs-all splitting unless labeled so explicitly, where c' is an offspring of c. This method enables relational shrinkage via data sharing.

[2]For webpages, `Byte4Gram` consumes far more extraction time than `Unigram` and could slow down our system in real-time classification especially for very long documents.

Table 3: Comparison of feature extractors, where `*-norm` denotes instance-level ℓ_1-normalization.

	Feature	Avg AUC
Tweet	`Byte4Gram`	baseline
Tweet	`Unigram`	-5.7%
Tweet	`Byte4Gram-norm`	+0.0%
Webpage	`Byte4Gram`	baseline
Webpage	`Unigram`	-0.1%
Webpage	`Unigram-norm`	+1.2%
Webpage	`Unigram-logTF-norm`	+1.5%

Table 4: Comparison of regularization type. ElasticNet, with automatic regularization strength tuning, improves overall model AUC.

	Regularization	Avg AUC	%topic win	Sparsity
Tweet	ℓ_2	baseline	baseline	baseline
Tweet	ℓ_1	-0.33%	36%	1.7%
Tweet	EN, λ^*	+0.98%	100%	5.1%
Webpage	ℓ_2	baseline	baseline	baseline
Webpage	ℓ_1	-0.23%	29%	1.4%
Webpage	EN, λ^*	+0.75%	100%	4.6%

Table 5: Using relational regularization significantly improve model quality.

	Method	Avg AUC	%topic win
Tweet	Flat	baseline	baseline
Tweet	Label Exp.	+3.6%	70%
Tweet	Cost Sens.	+3.3%	66%
Tweet	Hie. Reg.	+3.6%	69%
Webpage	Flat	baseline	baseline
Webpage	Label Exp.	+2.3%	79%
Webpage	Cost Sens.	+2.2%	73%
Webpage	Hie. Reg.	+2.2%	81%

Cost-sensitive learning The structure of the taxonomy indicates that we should penalize mistakes involving different topic pairs differently, e.g., misclassifying a "movie" tweet as "entertainment" should receive less cost than misclassifying it as "sports". This can be done by encoding the ontological relations into a cost matrix E, where the (c, c')-th element, $e_{cc'}$, represents the cost of misclassifying an instance of topic c into topic c' and optimizing the regularized expected cost $\mathbb{E}[e(y, \hat{y})|x)] = \sum_{c=1}^{k} e_{yc} p(\hat{y} = c|x)$, where we use as $e_{cc'}$ the tree-distance between c and c' in the taxonomy.

Hierarchical regularization [28, 15] We can encode the hierarchical dependency among topics into the regularization so that the model of a topic c is shrunk towards that of its parent node parent(c), e.g. by adding a penalty term, $\frac{1}{2}\eta \sum_{c=1}^{k} ||w_c - w_{\text{parent}(c)}||_2^2$, to Eq(1).

These approaches, while tackling relational regularization in three different aspects (i.e., data sharing, relational objective, parameter sharing), usually achieve equivalent effects. Note that "cost-sensitive optimization" is more versatile as it can also handle label dependencies that are discovered from data (e.g., topic correlations) rather than prior knowledge.

Experiment Results. In Table 5, we compared LR without relational regularization (denoted "Flat") vs the three approaches we described above. Relational regularization significantly improve model quality, e.g., over 2% boosts of average AUC. The differences among these three methods are nevertheless very small. For simplicity of implementation, hereafter, we primarily use label expansion.

4.6 Model calibration

LR models return *soft* probabilistic scores $p(y|x)$ in the range of $[0, 1]$. In order to apply them to label the topics of tweets/webpages, we need to calibrate the models towards specific quality criteria in need, i.e., a precision target in our case. There are two key questions here: (1) "over what distribution should precision be measured?" and (2) "how to find the optimal decision threshold?". For precision measurement, stratified sampling in the range of posterior scores produced by a classifier has been advocated as a way to reduce the variance of precision estimates [4]. While this method is very effective once the decision threshold are set, it is not straightforward to use when one seeks to determine the optimum thresholds.

Given a topic c, a small seed set of positive examples $\mathcal{P}_c$, an unlimited stream of unlabeled data $\mathcal{U}_c$ and a labeling oracle O, we use the following method to tune threshold θ_c:

Initial threshold Use $\mathcal{P}_c$ to estimate a rough lower bound θ_c^l for θ_c by using a much lower precision target.

Stratified sampling Apply the model to the output distribution and divide the interval $[\theta_c^l, 1]$ to equal-percentile bins[3]. Apply stratified sampling to $\mathcal{U}_c$ according to these bins and send the samples to O for confirmation labeling (Section 4.7), with results denoted $\mathcal{L}_c$.

Threshold estimation Let θ_j represent the posterior score of bin j, estimate the precision π_j for each bin j based on $\mathcal{L}_c$. The optimal threshold is the minimum breakpoint at which calibrated precision exceeds the precision target $\tilde{\pi}$, i.e.,

$$\boxed{\theta_c = \arg\min \theta, \quad \text{s.t.:} \ \frac{\sum_{\theta_j \geqslant \theta} s_j \pi_j}{\sum_{\theta_j \geqslant \theta} s_j} \geqslant \tilde{\pi}}$$

where s_j represents the size or prevalence of examples in bin j, which can be estimated using $\mathcal{U}_c$.

Note that, other than the above one-side calibration approach, a two-side approach is also possible, i.e., by initially narrowing down the range with both upper and lower bounds and applying golden section search to find optimal θ_c by progressively dividing bins (rather than fixed-size bins) and sub-stratified sampling. Although this two-side method can produce threshold estimations with arbitrary resolutions, we found in our experiments that one-side approach performs well enough and is much simpler to implement.

4.7 Quality evaluation

As we strive to achieve 90% or higher precision, high-quality gold standard data and accurate evaluation is critical to assess whether we achieve that goal. We use crowdsource human labeled data for quality evaluation. As we previously discussed in Section 4.2, human annotation is less accurate in a multi-label setting due to the *cognitive overload* caused by any nontrivial taxonomy. We instead ask for *confirmation labeling*. That is, rather than presenting the whole taxonomy to crowdsource workers and asking them to select whatever labels are relevant to a given tweet, we present a (tweet, topic) pair and ask them to provide binary answers to "whether or not the supplied topic is correct for the given tweet". As our primary goal is to estimate precision, binary judges on a sample output of our system are sufficient for our purposes. In principle, this protocol would require a lot more questions (i.e., labeling tasks) and in turn incur more costs if we were to estimate recall, e.g., for a given tweet and a 300-topic taxonomy, we need 300 labeling tasks in order to know the ground-truth topic of the tweet, compared to one task in the multi-label setting. Nevertheless, for precision assessment, this protocol has better quality assurance and it is more economical – because binary tasks are much easier, we found crowdsource workers are more willing to take the tasks at a much lower price, and more importantly, they are less likely to provide mistaking judgments.

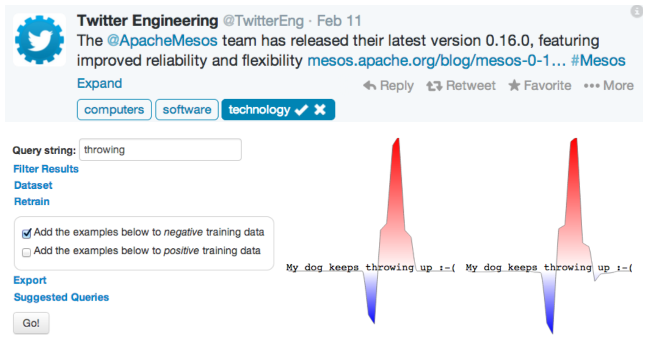
To effectively control cost, we assign each (tweet, topic)-pair sequentially to multiple workers; once the standard error of the responses is below a certain threshold, the label is considered final and the task will be frozen without being assigned further. The quality of the response varies worker by worker. To ensure quality, when assigning tasks, we also incorporate a small set of randomly-sampled probe tasks for which we already know the true answers with high confidence. Those workers whose response accuracy consistently falls below our requirement will not be admitted for future tasks, and their response in the current batch are ignored.

Recall estimation faces challenges due to the nature of our labeling protocol, the needs for unbiased corpus and the fact that the distribution of our data evolves over time, which is the subject of another paper [7]. Instead, we use primarily precision and coverage for quality evaluation.

4.8 Diagnosis and corrective learning

Once the system is deployed in production to label the topics of tweets, it is important to be able to identify cases where the model fails, and provide diagnostic insights as well as corrective actions to fix it on the flight. To this end, we instrumented a mechanism that associate topic labels as tags to tweets and expose them to the users. When users scroll on a topic tag, two buttons will show up to allow users to provide "right or wrong" feedback about the topic tag. Clicking on a topic tag will also take you to a channel consisting of tweets all about that topic, which is useful for diagnosis of a topic model as tweets are flowing through. The UI of this feature is shown by the top chart of Figure 5. This feedback collection module makes it easy to exploit the *wisdom of crowd* in an ad hoc way to receive instantaneous feedback on the performance of our topic models and provide opportunities to identify patterns for correction. Once failing cases or trouble areas are identified, another tool is used to visualize which parts of the tweet contributed to the offending decision the strongest, as shown in the bottom charts in Figure 5. This is useful to allow the modeler to manually zoom into the model and identify potential ovefitting patterns. With all the diagnostic patterns, we then employ *corrective learning* [25, 23] to correct the mistakes of an existing topic classifier either manually or automatically on the flight. Corrective learning is used because it is desirable to adjust models *gradually* using a relatively small

[3]The boundaries of the bins can be defined based on the percentiles of the posterior scores of $\mathcal{P}_c$ or $\mathcal{U}_c$.


Twitter Engineering @TwitterEng · Feb 11
The @ApacheMesos team has released their latest version 0.16.0, featuring improved reliability and flexibility mesos.apache.org/blog/mesos-0-1… #Mesos

Query string: throwing

[✓] Add the examples below to *negative* training data
[ ] Add the examples below to *positive* training data

My dog keeps throwing up :-(My dog keeps throwing up :-(

Figure 5: The diagnosis, corrective learning and feedback collection modules.Top: the feedback collection UI; Bottom: the corrective learning UI (Left), and visualization of an example tweet, which was misclassified as "rowing" before (Mid) but corrected after (Right) corrective learning.

number of new examples, and in such a way that the corrective action is limited to the area of the input space where the current models are actually misbehaving, while leaving the remainder unaffected. In Figure 5 (bottom), we show the UI for manual corrective learning, and one particular examples. The tweet was misclassified into the topic "Rowing" because of overfitting to the Byte4Grams extracted from the substring "-rowing", as shown in the bottom left chart (posterior score $p(c|x) = 0.97$). Fortunately, once we identify this pattern and adjust the model with corrective learning, this mistake can be corrected, as shown in the bottom right chart ($p(c|x) = 0.41$).

4.9 Model fine-tuning

Although we strive to turn down the noise in our labeled data by employing various techniques such as topic prior based pre-filtering, co-training and PU-learning, the data coming out of our training data acquisition pipeline still contains substantial noises. As a result, very few of our topic classifiers, if trained on these data, can achieve 90% precision. On the other hand, the labeled data collected from "feedback collection" and "quality evaluation" processes, albeit expensive and in relatively small amount, has very high precision as it is produced by human with controllable quality assurance. Can we use this small set of high-precision data together with the the large set of relatively noisy data to train better models? Naively training models on the combined data set is ineffective as only negligible AUC improvements were observed in our experiments (i.e., high-precision data is diluted/polluted by noisy data which is dominant in amount). We propose here a two stage training process:

Combaing good & noisy data directly not good

Pretraining Train LR model on the larger set of noisy data as described in Section 4.4.

Fine tuning Tune the model on the small high-quality data set *in the neighborhood of* the pretrained model.

To fine tune the model, we use the pretrained model, denoted w^0, as prior and shrink w towards w^0 while minimizing the regularized loss on the high precision data:

$$\boxed{\min_{w,b} \frac{\delta}{2}||w - w^0||_2^2 - \frac{1-\delta}{n}\sum_{i=1}^{n} l(w, b|x_i, y_i) + \lambda(\alpha||w||_1 + \frac{1-\alpha}{2}||w||_2^2),} \quad (2)$$

→ On high quality data

where $\delta \in [0, 1]$ controls how much the model w is shrunk towards its prior w^0 *vs* w^* (i.e., the maxima of the regularized likelihood on high-precision data), we sample δ from Beta($\delta|n_0, n$), where n is the number of examples seen in the training set and n_0 is a positive number quantifying how strong we trust the prior[4]. This way, the shrinkage adaptively finds a balance between the prior and the likelihood such that the more high-precision data we supply, the less the model will be shrunk towards the prior and vice versa.

Experimental Results. In Table 6, we report the quality of the models produced by the two-stage training process. As a reference, the very first *baseline*(i.e., the third row of Table 1) and the *stage-1* model (i.e., the third row of Table 5) are used for comparison. For more accurate assessment, we use crowdsource confirmation-labeled data (Section 4.7) in this evaluation and report the exact AUC scores. The fine-tuning (*stage-2*) consistently and significantly improves the quality of the model on both tweet and webpage, e.g., average AUC is improved by over 5% on tweets.

In Table 7, we also report the quality metrics of the calibrated models as seen in our service. The fine-tuned tweet model achieves ~90% precision with 33% tweet converge and over 80% topic coverage, a huge improvement from our first baseline, which, for example, with ~70% precision only achieves 8% tweet coverage and 21% topic coverage.

Table 6: Comparative evaluation of models on crowdsource confirmation-labeled data.

	Model	Avg AUC	%topic win
Tweet	baseline	0.7292	–
Tweet	stage-1	0.8004	89%
Tweet	stage-2	0.8415	97%
Webpage	baseline	0.7496	–
Webpage	stage-1	0.7845	85%
Webpage	stage-2	0.8051	99%

Table 7: Quality metrics of the calibrated classifiers.

	precision	coverage	topic coverage
Tweet	89.7%	33%	81%
Webpage	91.2%	53%	73%

4.10 Discussions

Large-scale data. We discuss here two approaches, enabled by the prior shrinkage framework[5], which scale up our learning capability to massive data set in terabytes:

- **Sequential mini-batch learning.** With this approach, we partition the data set into multiple small segments $\{\mathcal{D}_1, \ldots, \mathcal{D}_m\}$ and train the model on each

[4]The best choices of n_0 and n can be decided by applying Laplacian approximation to the likelihood. We use cross-validation in our experiments for simplicity.

[5]Note that hierarchical regularization is also enabled by prior shrinkage.

Fine-tune periodically

partition sequentially with prior shrinkage, i.e., suppose w^k is the model trained on $\mathcal{D}_k$, w^{k+1} will be trained on $\mathcal{D}_{k+1}$ with w^k as prior. Compared to the two extremes, i.e., pure streaming learning and single-thread batch-mode learning, this approach provide a good trade-off between computational and statistical efficiency (i.e., it takes less time than batch-mode and requires few examples to converge than streaming)[10]. Moreover, when the quality of the data is progressively improved in the partitions (e.g., sampled and labeled via active learning), we found this approach can progressively improve the quality of the model.

Active Learning

- **Distributed batch-mode learning** This approach is populated with the recent advances in operational research on *alternating direction method of multipliers* (ADMM) [8]. The basic idea is to maintain a global consensus $\bar{w}$ and iterate between (1) maximizing the (unregularized) log likelihood on each partition $\mathcal{D}_k$ with shrinkage towards $\bar{w}$, to obtain a local solution w_k, usually done in a distributed manner or in parallel, and (2) updating the consensus $\bar{w}$ to minimize the regularization penalty and the deviations from the local solutions $\{w_k\}$, which is usually very cheap to obtain (e.g., close-form solution in our case).

Active learning. One question one may ask is: since high-precision human labeled data is so effective in improving the model quality yet it is so expensive to acquire, would it be more efficient to apply active learning, *i.e.*, to train the model from scratch with a minimum amount of iteratively acquired training data? Unfortunately, the answer is "no". First, active learning requires integration of our machine learning capability into the API of the crowdsource agent, which is not supported by any crowdsouring services on the market. More importantly, the feature vector for each tweet is extremely sparse (e.g., only ~70 binary occurrence in a 1M space). At such sparsity, even if we were to see each feature to occur once (for each of the 300 topics), it would require millions of tweets for querying and incur a cost that easily exceeds our budget. In contrast, we find corrective learning (Section 4.8) is more suitable to our case – it provides useful diagnosis and corrective feedback to an (underperforming) existing model, and allows us to quickly fix it on the flight; moreover, it also takes advantages of the large user base of Twitter to acquire high-quality labeled data at virtually no cost by using the wisdom of the crowd.

Active Learning didn't suit their task. But could be useful.

5. BEYOND TEXT CLASSIFICATION: INTEGRATIVE TOPIC MODEL

Up to now, the topic models we present infer topics solely based on the text of tweets. However, a tweet is an envelope of different things, for example, besides (1) the text, a tweet can also contain (2) author, (3) embedded URL, (4) entities (e.g., `#hashtag`, `@mention` and named entities), (5) engagers who have interacted with the tweet (e.g., favorite, retweet, reply), (6) multi-media (e.g., image, video) and (7) other context information such as geographic, temporal and social contexts. The question is, how to harness all these sources of signals to achieve highly accurate topic inference when each single signal might fall short?

One approach is to extract feature from each source of signals and train LR models in the augmented feature space. There are some drawbacks with this approach. First, not all the features are available for every tweet (they may available at different stages or not available at all, e.g., engagements), leaving a lot of missing values. Second, the significant increase in feature dimensionality raises a lot of complications, e.g., it requires more labeled data for training and incurs significant overhead in memory consumption and runtime latency. Experiments also suggest that this approach is less effective in our system. An alternative approach is to use a hierarchical model similar to a *feed-forward neural network*, e.g., train a set of LR models on each input signal alone, and another layer of LR model on top of the output of these models. The structure doesn't have to be three layers. If needed, more hidden layers can be used to capture higher-order correlations among different features to further improve generalization performance as suggested by recent advances in neural network research [16]. Unfortunately, this model requires more complicated machine learning capabilities which are not currently supported by our infrastructure.

Limitations in 2014.

Instead, we present here an integrative inference approach based on *decision aggregation* (DA), where each of the signals is used as a weak decision maker to provide noisy topic labels for tweets, and the final topic labels are decided by aggregating these noisy predictions, e.g., via weighted majority voting. There is no restriction on what types of predictors to be used by each weak predictor as long as they all produce predicted labels for a given tweet. Similar methodologies have been applied successfully in the literature to handle tasks that involve multiple noisy experts or highly heterogeneous data [11, 24]. In this section, we first describe how we derive topic decisions from users and entities, we then present the integrative inference algorithm in details.

5.1 User topic model

A user is usually interested in only a few topics out of the 300+ topic space. When associated with a tweet via authorship or engagement, the user's interest topics provide a valuable signal for inferring the topic of the tweet. User interest modeling is itself an important topic. In fact, one primary use case of tweet topic modeling is to understand the interest intent of Twitter users so as to enable topic-based personalization.

Based on what a user has *produced* and *consumed* on Twitter, we derive two types of interest topics. Particularly, if a user u prominently tweets about a topic c, we call u is "*Known-for*" (KF) c. Likewise, if a user u consistently shows interest to tweets about topic c as indicated by his engagements (e.g. retweet, favorite, reply, clicks), we say u is "Interested-in" (II) c. Note that KF of u is a subset of u's II. In our system, the distribution of interest topics for each user (i.e., KF and II) are derived in real-time by running a *tfidf*-style topic-ranking algorithm on a sliding widow. The *foreground* model is a smoothed likelihood of user u producing / consuming tweets about topic c, whereas the *background* model is a population-level mean of the former. Let $r_t(c|u)$ denote the *tfidf* score of topic c to user u at time t, the topic distributions are rolled-over with a time-decay weight via $p_t(c|u) \propto \frac{1}{K(t)}\int_0^t k(t-s)\phi(r_s(c|u))ds = (k(1)p_{t-1}(c|u) + k^2(0)\phi(r_t(c|u)))/(k(1) + k^2(0))$, which is then normalized (over all the topics) and memcached for real-time access. Here $k(\cdot)$ is a time-decay kernel (e.g., the Exponential PDF $k(t) = \beta e^{-\beta t}$), $K(t) = \int_0^t k(s)ds$ is the CDF of $k(t)$, ϕ is a feature transformation function (e.g.,

Table 8: User interest models are evaluated in terms of precision at n (P@n) on user survey data.

Definition of positive interest	P@1	P@2	P@3	P@5
somewhat interested & above	0.81	0.85	0.85	0.85
interested & above	0.69	0.71	0.73	0.73

when $\phi(x) = \exp(x)$ is used, the distribution $p(c|u)$ will be a weighted softmax transformation of $r_t(c|u)$).

Another source of information for deriving user interest topics is to leverage the *social annotation* mechanism enabled by Twitter List. In particular, if a user u is *consistently* labeled by other users with lists that can be mapped to topic c, we call u is known-for c. II topics are derived from KF by using the follow graph, i.e., if a user u directly follows a user v who is unambiguously known-for topic c, we say u is interested-in c.

Interest topics derived from authorship and engagements are dynamic, and can adapt over time as users' interests evolve. In contrast, interests derived from social annotation are static. Nevertheless, we found that, especially for celebrities who don't tweet proactively, social annotation is more accurate. In our system, these two approaches are combined linearly with weight tuned on validation data.

Experimental Results. To evaluate the derived user interest topics, we conducted an online survey to collect explicit feedback from users. For each participating user, we select a set of $\sim$ 10 topics, consisting of a subset of topics predicted by our model and a few randomly-sampled probe topics. We present these topics to users in a random order and ask them to rate how much they are interested in tweets on each topic. Users are allowed to provide their answers on 7 scale from "strongly interested (7)" to "strongly uninterested (1)", and one additional option of "not sure". Irresponsible or noisy answers which apparently conflict with what users explicitly stated in their profiles or other strong behavioral signals were filtered to reduce the noise. Based on $\sim$ 5000 survey responses, we assess our model in terms of *precision at the top-n* topics (or P@n). The results are reported in Table 8. The average score of the top-5 predicted interest topics is around 6 (i.e., interested), significantly higher than the average score of the probe topics, which is around 3 (i.e., somewhat uninterested), indicating that the predictions by our model are consistent with what users perceive about their true interests. Overall, our model achieve up to 80% precision, and on 34% tweets, the top-5 predicted interest topics overlap with the ground-truth interests of a user.

5.2 Hashtag topic model

We present here an algorithm to infer topics from `#hashtags`, although the algorithm is applicable to other types of entities (e.g., named entities, `@mentions`) as well. Hashtag is one of the key features that define Twitter. In its essence, "*hashtag* = *topic*", i.e., hashtags are user-defined topics to better organize conversations. Hashtags are also way more predictive than ordinary text strings. Most hashtags span a rather narrow bandwidth out of a large spectrum of topics, e.g., `#ObamaCare`, `#TheVoice` and `#NBA`.

A straightforward approach is to use hashtag as features and train a LR model to classify tweet solely based on hashtags. This approach is, however, inherently flawed due to the huge volume (e.g., the number of hashtags are in $O(100Ms)$), the transient nature (i.e., hashtags are continuously evolving over time with new hashtags emerging every second) and the extreme sparseness (i.e., the majority of tweets usually contain no more than one hashtag). Indeed, our experiments indicate that, when hashtags are used as special unigram features together with regular text strings, only a very marginal (on some topics even negative) improvements were observed.

Table 9: Performance of hashtag topic model: precision at n (P@n), tweet coverage, and the ratio of output overlap with tweet classifier.

P@1	P@2	P@3	P@5	coverage	overlap ratio
32%	49%	57%	64%	6.6%	53%

Instead, we infers hashtag topics via a retrieval algorithm, i.e., take a hashtag as a query, we aim to find the most representative topics for it. Unfortunately, because the hashtag-to-topic retrieval task is very different from conventional query-to-document retrieval, standard retrieval models such as pointwise mutual information (PMI), $tfidf$, BM25 and χ^2-test did not perform well in our experiments. To this end, we present a *learnable* retrieval model supervised by the output of our high-precision tweet text classifiers.

Feature For hashtag h and topic c we use the following

$$p(h|c), p(h|\neg c), p(c|h), p(c|\neg h)$$

and their corresponding complement probabilities, together with the occurrences $p(h)$ and $p(c)$ as features, where $\neg a$ means the absence of a. Standard retrieval models such as PMI, $tfidf$, BM25 and χ^2 are all prescribed functions of some subsets of these 10 features.

Model We train a linear model using the logarithms of the aforementioned features: $r(c|h) = \langle\theta, \log(f)\rangle$. The parameter θ is tuned such than a retrieval metric (e.g., nDCG) is maximized on the training subset of high-precision tweet classifcation results.

Note that we can turn the direction of the retrieval model around, i.e., to find the most predictive hashtags for each topic, which is useful for a number of features such as hashtag-based topic summarization, topic-aware trends, as well as topic priors for training data acquisition (Section 4.2).

Experimental Results. In evaluation, we randomly sampled $\sim$ 500 hashtags with 10 example tweets for each hashtag, and asked crowdsource workers whether the ground-truth topic is within the top-n topics retrieved from hashtag. The results are reported in Table 9. Overall, the model achieves 64% precision at position 5 (compared to 34% of user II topics). Moreover, hashtag topics overlap with tweet classification results on only 53% of the tweets, suggesing that they are also potentially useful for improving recall.

5.3 Integrative inference

We now have five topic predictors for a tweet, i.e,

- **Tweet text**: classification based on tweet content
- **Webpage**: classification of webpage (embedded URL)
- **Author KF**: known-for topics of the author,
- **Hashtag**: topics from the embedded `#hashtags`,
- **Engager II**: aggregated interested-in topics of the users who engaged with the tweet.

Table 10: Quality metrics of integrative inference.

Precision	Tweet Coverage	Topic Coverage
93%	37%	81%

Table 11: Importance ranks and scores of the 5 ingredient predictors in integrative inference.

1	2	3	4	5
Tweet text	Webpage	Hashtag	Author KF	Engager II
1.0	0.92	0.68	0.55	0.47

We derive the final topics for each tweet by aggregating topic labels provided by each of these predictors via *decision aggregation*. In settings where there are multiple experts providing noisy (possibly conflicting) decisions yet none of them is good enough alone, decision aggregation has been shown to beat or perform comparably to the best expert in hindsight [11, 24]. As we are primarily concerned with precision, we aggregate the noisy topic labels providing by each predictor using *weighted majority voting*: we assign a weight for each predictor and take all the topic labels to form a committee to vote, where each topic takes the sum of weights from the predictors who activate it. These topics together with their weights are then propagated according to the taxonomy tree to arrive at the final topic predictions. In our experiments, the weights are trained using the *AdaBoost* algorithm [14] on human confirmation-labeled data.

The integrative algorithm brings a close-loop inference mechanism and enables our system to dynamically adapt itself to cope with data drift, an importantly ability that a static LR classifier lacks. This is due to the fact that, by doing integrative inference, our system is running in real-time a close-loop iteration between (1) inferring topics about entities and users from tweet topics and (2) refining tweet topics using entitiy and user topics based on emerging signals.

Experimental Results While we can configure the integrative inference algorithm to improve precision and coverage (or recall) at the same time, here we exemplify with a rather conservative configuration to optimize precision (i.e., eliminate maximum false positives). The results are depicted in Table 10. At 12% higher coverage (compared to the tweet classifiers), the integrative model achieves 93% precision. Table 11 also shows the rank and importance score (normalized against the maximum score) of the 5 ingredient signals, as learned by the AdaBoost algorithm.

6. SUMMARY

We have presented a deployed large-scale topic modeling system that infers topics of tweets over an ontology of hundreds of topics in real-time and at stringently high precision. We have proposed a unique collection of topic modeling techniques that effectively helped us to address the challenges in implementing the system and satisfying the quality requirement.

Acknowledgements. We would like to thank our interns Ramnath Balasubramanyan, Mitja Trampus, Naftali Harris and Will Landecker, and the other members on the user modeling team especially Matthew Bilotti, Jake Manix, Praveen Bommannavar, Dimitar Simeonov, Dong Wang and David Burkett for their contributions to the system.

7. REFERENCES

[1] `http://about.twitter.com`.
[2] `https://blog.twitter.com/2013/new-tweets-per-second-record-and-how`.
[3] R. Balasubramanyan and A. Kolcz. Chatter in twitter: Identification and prevalence. In *ASONAM*, 2013.
[4] P. N. Bennett. Using asymmetric distributions to improve text classifier probability estimates. In *SIGIR*, 2003.
[5] D. M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, Apr. 2012.
[6] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, 1998.
[7] P. Bommannavar, A. Kolcz and A. Rajaraman. Estimating recall for rare topic retrieval via conditionally independent classifiers. paper pending review, 2014.
[8] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, Jan. 2011.
[9] M. Burgess, A. Mazzia, E. Adar, and M. Cafarella. Leveraging noisy lists for social feed ranking. *ICWSM*'13.
[10] V. Chandrasekaran and M. I. Jordan. Computational and statistical tradeoffs via convex relaxation. *PNAS*, 2013.
[11] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World.* Cambridge University Press, 2010.
[12] C. Elkan and K. Noto. Learning classifiers from only positive and unlabeled data. In *KDD*, 2008.
[13] G. Forman and E. Kirshenbaum. Extremely fast text feature extraction for classification and indexing. *CIKM*' 08.
[14] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT*, 1995.
[15] S. Gopal and Y. Yang. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *KDD*, 2013.
[16] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 2006.
[17] L. Hong and B. D. Davison. Empirical study of topic modeling in twitter. In *SOMA*, 2010.
[18] J. Lin and A. Kolcz. Large-scale machine learning at twitter. In *SIGMOD*, 2012.
[19] J. Lin, R. Snow, and W. Morgan. Smoothing techniques for adaptive online language models: Topic tracking in tweet streams. In *KDD*, 2011.
[20] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval.* 2008.
[21] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013
[22] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *EMNLP*, 2009.
[23] K. Raman, K. M. Svore, R. Gilad-Bachrach, and C. Burges. Learning from our mistakes: Towards a correctable learning algorithm. In *CIKM*, 2012.
[24] V. C. Raykar, S. Yu, L. H. Zhao, A. Jerebko, C. Florin, G. H. Valadez, L. Bogoni, and L. Moy. Supervised learning from multiple experts: Whom to trust when everyone lies a bit. In *ICML*, 2009.
[25] B. Settles. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *EMNLP*, 2011.
[26] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. *ICML*' 07.
[27] W. X. Zhao, J. Jiang, J. Weng, J. He, E.-P. Lim, H. Yan, and X. Li. Comparing twitter and traditional media using topic models. In *ECIR*, 2011.
[28] K. Zhou, S.-H. Yang, and H. Zha. Functional matrix factorizations for cold-start recommendation. *SIGIR*' 11.