Smoothing Techniques for Adaptive Online Language Models: Topic Tracking in Tweet Streams

Jimmy Lin, Rion Snow, and William Morgan
Twitter
795 Folsom St.
San Francisco, California
@lintool @rion @wm

ABSTRACT

We are interested in the problem of tracking broad topics such as "baseball" and "fashion" in continuous streams of short texts, exemplified by tweets from the microblogging service Twitter. The task is conceived as a language modeling problem where per-topic models are trained using hashtags in the tweet stream, which serve as proxies for topic labels. Simple perplexity-based classifiers are then applied to filter the tweet stream for topics of interest. Within this framework, we evaluate, both intrinsically and extrinsically, smoothing techniques for integrating "foreground" models (to capture recency) and "background" models (to combat sparsity), as well as different techniques for retaining history. Experiments show that unigram language models smoothed using a *normalized* extension of *stupid backoff* and a simple queue for history retention performs well on the task.

Categories and Subject Descriptors

H.1 [**Models and Principles**]: General

General Terms

Algorithms, Measurement, Performance

Keywords

Twitter, TDT, stream processing

1. INTRODUCTION

The rapid growth of real-time, high-volume communication on services such as Twitter has led to interesting new challenges and opportunities in tracking consistent topics across real-time streams of messages. This paper tackles the problem of tracking topics in a continuous stream of short natural language texts. Our specific application is the popular microblogging service Twitter, whereby users can send short, 140-character messages, called "tweets", that are distributed to "followers". As of Spring 2011, Twitter has over 200 million users who collectively create over one billion tweets each week. Whenever major news events happen, the service routinely achieves over 4000 tweets per second.

Twitter is worthy of study from many different perspectives: its length constraint makes it distinctive in terms of language use, with substantial differences from typical texts studied by researchers in natural language processing and information retrieval such as newswire text. Tweets provide a window into users' behavior, both online and in the physical world, and provide a second-by-second update of "what's happening" in the world. Tweets capture everything from the mundane daily routines of the masses (i.e., "what I ate for breakfast") to spectacular breaking events to other messages of significant historic impact. When a US Airways jet made an emergency landing in the Hudson in January 2009, the world learned of the news via a tweet. When Iranian authorities clamped down on media coverage following the disputed presidential election in 2009, news leaked out via tweets. More recently, Twitter has been credited as an important tool that aided Egyptian protesters in toppling the Mubarak regime in February 2011—serving both as a communications platform for grassroots organization and a channel for information dissemination when other outlets were shut off. Mainstream news organizations such as CNN already consider Twitter an indispensable journalistic tool, and tweets are regularly featured on news broadcasts. Finally, researchers have shown that tweets provide a probe into the sentiments and behaviors of entire populations, enabling what has come to be known as computational social science. To give just two examples: researchers have found that tweet sentiment correlates with public opinion polls [17], and that tweets can be used to predict opening weekend box office revenues for movies [2].

Given a continuous stream of incoming tweets, we are interested in filtering and retaining only those that are relevant to a particular topic such as "American football", "Apple", and "fashion". We assume that at any given moment, there are a relatively small number of such topics (on the order of a few dozen), and for the sake of simplicity, we assume that these topics are specified in advance (for example, by editors who wish to curate the tweet stream). The problem setup naturally fits a streaming model of computation [16], in which events arrive continuously at a high rate and need to be processed in bounded space and time. An event is only encountered once and subsequently "lost" forever. Although an algorithm can "remember" previously seen events, it is assumed that the total amount of data is significantly larger than the space (i.e., memory) available to the algorithm.

The high arrival rate of events in typical streaming settings limits the amount of processing that is practical (at 1000 messages per second, a system has only one millisecond to process the message).[1] Therefore, one common approach to such problems is to

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
KDD'11, August 21–24, 2011, San Diego, California, USA.
Copyright 2011 ACM 978-1-4503-0813-7/11/08 ...$10.00.

[1] Of course, we recognize that distributed stream-processing frameworks exist (e.g., [8]). However, moving to a multi-machine setup inevitably complicates the architecture from an engineering per-

eschew sophisticated models and start from the simplest technique that can possibly work. In this paper, we use simple language models (LMs) that are trained using hashtags contained in a small fraction of tweets, which serve as proxies for topic labels. These language models are adaptive in the sense that newly encountered tweets update the probability distribution in an online fashion. The LMs are then applied to filter the entire tweet stream by computing perplexity on unseen tweets: by varying the decision threshold, we can realize different precision/recall tradeoffs.

Within this general framework, we explore the design space in two dimensions:

1. different smoothing techniques for integrating "foreground" models (to capture recency) and "background" models (to combat sparsity); and,
2. different techniques for retaining history to ensure that the algorithm runs in bounded space.

We conducted both intrinsic evaluations as well as extrinsic evaluations in a topic tracking task. We find that stupid backoff [5]—an exceedingly simple smoothing technique—with a relatively short event queue works well.

The remainder of the paper is organized as follows: We begin with a discussion of related work in Section 2. Our general approach is laid out in Section 3. Data are described in Section 4, followed by results of evaluations: intrinsic (Section 5) as well as extrinsic (Section 6). We conclude in Section 7.

2. RELATED WORK

Although our problem is superficially similar to the goals of the TDT (topic detection and tracking) program [1], there are important differences. Topics in TDT are defined "to be a set of news stories that are strongly related by some seminal event that triggers the topic". Instead, we are interested in a broader definition of topics that might include a disparate collection of loosely-related events. To illustrate by example, a topic in the TDT sense might be the event "San Francisco Giants win the World Series", whereas we would define a broader topic around "baseball". Of course, the contents of the baseball topic would evolve over time and likely include the World Series story around early November 2011. Thus, topic tracking in our sense can be interpreted as the answer to the question "What's happening right now?"

Analysis of social media has attracted a lot of attention from the research community in recent years. Much work has specifically focused on Twitter, much more than can be adequately covered here. Examples include descriptive characterizations [12, 6], sentiment analysis [10, 17], entity tagging [15], influence modeling [22, 3, 4], topic modeling [19], conversation modeling [20], and modeling the diffusion of information [13, 21]. Most relevant to this study is the work of Petrović et al. [18], who examine the first-story detection problem in tweet streams. However, they adopt the narrow TDT definition of an event (i.e., a specific news event), as opposed to our broader conception, and therefore are addressing a fundamentally different problem (with a different technical approach).

Another thread of related work is the growing interest in streaming algorithms in natural language processing [14, 9, 18], highlighting the importance of time- and space-efficient algorithms in dealing with modern problems. Our work builds on this thread and contributes to this emerging area.

3. APPROACH

Our goal is to predict the topics of interest associated with particular tweets. To make this problem more tractable, We assume that these topics of interest are denoted by hashtags, which are user-specified labels that begin with the hash character (#). It is interesting to note that hashtags were not originally part of Twitter, but evolved organically as a tool for users to organize their tweets. For example, a tweet about American football might be "annotated" with the hashtag #nfl (which stands for the National Football League, the major organization of the sport):

> Jay Glazer reports Vikings, Patriots are indeed talking trade for WR Randy Moss http://tinyurl.com/2uhbucb #NFL

Approximately a tenth of tweets contain one or more hashtags. They represent a bottom-up emergent phenomenon, as there is no authoritative source prescribing their use (or misuse). Indeed, many hashtags emerge spontaneously as cultural memes—although that process is beyond the scope of this work. Here, we focus on "persistent" hashtags that represent coherent, unambiguous topics that are (relatively) stable over time. For the sake of simplicity, we assume that the hashtags of interest (i.e., those topics we wish to track) are provided in advance. We expect that across all of Twitter, there are on the order of a few dozens of such topics, so it is not unreasonable to expect editorial (i.e., human) curation in selecting relevant hashtags.

Even though our topics of interest are generally coherent and stable over time, topical tweets may be about vastly different people, events, or entities. For example, at some point in time for the #nfl topic, Twitter users may be discussing a team that is on a long winning streak, but a short time later the focus may shift to a player who has just been traded from one team to another (as above). As another example, a topic such as #fashion is (perhaps by definition) rapidly changing.

Our approach to the topic tracking problem is to build language models (LMs) for each topic, and then apply these topic LMs to filter the entire tweet stream (including the majority of tweets that do not contain any hashtags). With language models, we explored the simplest possible classifier—based on computing the perplexity of unseen tweets. By varying the decision threshold, we can realize different tradeoffs between precision and recall. Of course, our language models need to be *adaptive* since the content of tweets for a particular topic may change rapidly. Furthermore, the language model must adapt in an online streaming fashion, since tweets are continuously arriving at a rapid rate.

The fact that topics continuously evolve introduces a tension: on the one hand, to obtain accurate term counts (that are reflective of the present state of the world) we would only want to keep track of recent tweets. However, this introduces a sparsity problem, where a large part of the event space is never encountered—leading to the well-known problem of zero-probabilities. The solution is of course *smoothing* [7, 11], whereby some probability mass is transferred from seen to unseen events: we adopt the strategy of smoothing a "foreground" model, which tracks recent term counts, with a "background" model, which provides estimates of term distributions across a much longer span of time. We empirically examine the effectiveness of different smoothing techniques (discussed in Section 3.1). The other major design decision is how to retain "history" in the foreground language model so that the algorithm will run in constant space; since vocabulary size grows unbounded with increasing amounts of text (Heap's Law), at some point in time an algorithm *must* "forget" some data. Section 3.2 discusses simple techniques for retaining history.

spective. Instead, we focus on single-machine solutions that can be trivially replicated for robustness.

3.1 Smoothing Techniques

In this study, we explored both unigram and bigram language models. This section focuses on unigram language models (which are later shown to be more effective), but the description applies to bigram language models as well with standard straightforward extensions. Salient differences between the two are specifically noted.

Our basic approach involves smoothing a foreground language model with a background language model computed over a much longer period of time (one month in our case; see Section 4 for details). The purpose of the background model is to combat sparsity, since many terms may not have been observed in the foreground model (which retains limited history). In the unigram LM case, we denote the probability of a word given the background model as $P_{\mathcal{B}}(w)$. We estimate this distribution using the maximum likelihood criterion in an offline, batch setting. To restrict the vocabulary space, we discard all terms that occur ten or fewer times. The background model is itself smoothed using absolute discounting [11] (with $\delta = 0.5$). For the bigram case, we discarded bigrams that occur three or fewer times; the bigram LM was similarly smoothed using absolute discounting. For the unigram LMs, we discarded stopwords, but not for the bigram LMs. No stemming was applied in either case.

The foreground language model is defined in terms of a history, which specifies the policy under which recently-encountered terms (or bigrams) are retained. Section 3.2 describes specific history retention techniques in more detail. We use $c(w;h)$ to denote the count of word w within the history.

In this paper, we explored four different smoothing techniques to integrate the foreground and background models. Note that we excluded several well-known methods which are computationally expensive, for example, Katz backoff and Good-Turing estimation [7, 11], both of which require keeping track of frequencies of frequencies. The four smoothing techniques examined are as follows:

Absolute Discounting, where we subtract a partial count δ from each observed word and distribute the leftover probability mass to the background model. This is defined as follows, where w_n is the number of unique terms in the foreground model:

$$P(w) = \frac{\max(c(w;h) - \delta, 0)}{\sum_w c(w;h)} + \frac{\delta \cdot w_n}{\sum_w c(w;h)} P_{\mathcal{B}}(w)$$

Jelinek-Mercer smoothing, which involves a linear interpolation between the foreground and background models, parameterized by parameter λ, which controls the influence of each model:

$$P(w) = \lambda \cdot \frac{c(w;h)}{\sum_w c(w;h)} + (1 - \lambda) \cdot P_{\mathcal{B}}(w)$$

Bayesian smoothing using Dirichlet priors. A unigram language model is a multinomial, for which the conjugate prior is the Dirichlet distribution. This is defined as follows, parameterized with μ:

$$P(w) = \frac{c(w;h) + \mu \cdot P_{\mathcal{B}}(w)}{\sum_w c(w;h) + \mu}$$

Stupid backoff [5], which has been shown to be simple and effective for very large language models:

$$S(w) = \begin{cases} \frac{c(w;h)}{\sum_w c(w;h)} & \text{if } c(w;h) > 0 \\ \alpha \cdot P_{\mathcal{B}}(w) & \text{otherwise} \end{cases}$$

That is, if we have seen the term in the foreground model, we use its maximum likelihood estimate; otherwise we take the probability from the background model, scaled by α. Brants et al. [5] demonstrated that its simplicity allows the technique to be applied to larger corpora than practical with state-of-the-art techniques such as Kneser-Ney, and actually outperforms it at scale.

Note that stupid backoff does not define a valid probability distribution, which makes the score S not comparable to the other smoothing techniques. To remedy this we introduce a simple extension, *normalized* stupid backoff:

$$P(w) = \begin{cases} \frac{1}{1+\alpha} \cdot \frac{c(w;h)}{\sum_w c(w;h)} & \text{if } c(w;h) > 0 \\ \frac{\alpha}{1+\alpha} \cdot P_{\mathcal{B}}(w) & \text{otherwise} \end{cases}$$

In essence, we can renormalize stupid backoff scores into probabilities by dividing by the sum of scores. This has the nice effect of retaining all ranking properties of the scoring function.

3.2 History Retention Techniques

Orthogonal to the smoothing techniques discussed in the previous section is the method for retaining history (i.e., recently-encountered terms). In this work, we explored the following three techniques:

"Forget", the simplest possible approach, is to flush all data structures after every h events. That is, we periodically forget everything that has been observed (i.e., reset counts of all terms to zero).

"Queue" involves retaining the most recent h events in a FIFO queue. When the queue reaches capacity, newly encountered terms displace the earliest terms and counts are adjusted accordingly. This is equivalent to a fixed-width sliding window on the tweet stream. Although this technique occupies constant space, it requires remembering the entire sequence of events (as opposed to a simple counter in the "forget" approach).

"Epoch" is the approach described by Goyal et al. [9] for estimating approximate *n*-gram frequencies in a streaming setting. The basic approach is to delimit the incoming stream in terms of epochs (h events); after each epoch, low frequency *n*-grams are removed in a manner that provides theoretical guarantees on the true frequency of the remaining *n*-grams (see cited paper for more details). Note this is technically *not* a bounded space algorithm, although space requirements grow relatively slowly (logarithmically with respect to input size).

4. DATA

Our dataset consists of the raw (unfiltered and unsampled) tweet stream from October 1 to October 7, 2010 (inclusive) which spans an entire week. During this span, we observed an average of 93.5 million tweets per day. On average, 10.9 million tweets per day contain at least one hashtag. We manually selected 10 topics based on popularity and to obtain broad coverage of different types: sports, technology, popular culture, and general interests. These topics are shown in Table 1. Based on our experience, they represent a reasonably accurate sample cross section of typical Twitter users' interests. We discarded tweets that contained fewer than ten content words, not counting hashtags and usernames. Retweets, where users "pass on" tweets to their followers verbatim, are also discarded. The second column shows the number of tweets (that meet the above criteria) for each topic during the week span. For the background model, we used tweets from the entire month of September (2.66 billion tweets). In all cases we made no effort to perform

Topic	Tweets	Notes
1. #nfl	58913	American football
2. #mlb	39903	baseball
3. #apple	75696	Mac, iPhone, iPad, etc.
4. #android	45639	i.e., mobile phone
5. #glee	75174	TV show
6. #jerseyshore	25336	TV show
7. #teaparty	38099	political movement
8. #travel	34436	
9. #fashion	22743	
10. #health	51483	

Table 1: Topics explored in our study.

language identification and made no effort to separately tokenize or otherwise process non-English tweets.

5. INTRINSIC EVALUATION

Our first set of experiments involved intrinsic evaluation of the approach discussed in Section 3. The experimental procedure was as follows: the algorithm is initialized with the background model and an empty foreground model. Each tweet is processed in turn; tweets that do not contain the relevant hashtag or do not meet the content length requirements are discarded. For each tweet, we then remove the hashtag and compute its perplexity with respect to the current topic model, defined as:

$$\text{pow}\left[2, -\frac{1}{N}\sum_{i=1}^{n}\log_2 P(w_i)\right]$$

where the test tweet W is composed of N words $(w_1, w_2, \ldots w_n)$. Under the standard interpretation, a perplexity of *k* means that one is as surprised on average as guessing from *k* equiprobable choices at each symbol in the sequence. Finally, the foreground model is updated. This process is repeated until all tweets are consumed.

Table 2 presents the average perplexity across each topic using unigram language models under a variety of settings. We ran a large number of experiments exploring the parameter space; for space considerations, we only present the most interesting results.

The first row in the table shows the perplexity of the background model by itself, which provides the baseline. The first two blocks of the table show different smoothing techniques under the queue history retention technique ($h = \{1000, 10000\}$). The smoothing parameters represent effective settings derived from a coarse exploration of the parameter space. For Jelinek-Mercer, we found that $\lambda = \{0.3, 0.4\}$ works well in balancing the need to capture recency and to combat sparsity. With lower values we get too much influence from the background model, but with higher values we suffer from the inability to characterize rare or unseen terms in the foreground. It makes sense that a longer history calls for less weight on the background model. We noticed the following general effect when varying μ in Dirichlet smoothing: models with values too small suffer from sparsity, while those with values too large overemphasize the background model. Longer histories require a larger μ, because otherwise we are not receiving sufficient contributions from the background model. Interestingly, for absolute discounting we found that the best δ is somewhat higher than values typically reported in the literature. We explain this as follows: recall that in absolute discounting δ partial count is subtracted from every seen event; since we're only retaining limited history, the foreground vocabulary space is rather small—hence, we need a large discount to "save" enough probability mass for the background. Overall, Jelinek-Mercer appears to perform the best in terms of perplexity, although Dirichlet smoothing and absolute discounting are comparable for some topics. In nearly all cases, normalized stupid backoff is not competitive in terms of our intrinsic evaluation.

Comparing the two history sizes, it is interesting that while $h = 10000$ yields lower perplexities, we still obtain respectable perplexity values with only a tenth as much history. For certain space-constrained applications, this is a tradeoff worth considering.

In terms of absolute values, we observe large differences in perplexity, which gives an indication of how "cohesive" a topic is—and as we shall see later, this lets us quantify topic difficulty in the extrinsic task. We are able to build models with relatively low perplexity for topics such as #apple because the vocabulary is relatively restricted in terms of tweet content: terms such as iPad, iPod, etc. appear with high frequency. At the opposite extreme, a topic like #fashion is far less predictable in terms of tweet content (which makes sense).

The reduction in perplexity from adding a foreground model (compared to using only the background model) gives us a sense of how rapidly a topic changes over time. For a topic like #nfl, we observe a large decrease in perplexity—this make sense since "what's happening" in American football changes from moment to moment (e.g., successive stories on player trades, scouting reports, injury updates, etc.). At the other end of the spectrum, for a topic such as #apple there is a steady stream of discussion about popular Apple products (iPad, iPod, iPhone, etc.). There were no major new Apple products or announcements during the time considered in our study, so the content of tweets didn't differ much from the general discourse captured by the background model. Note that the "cohesiveness" of a topic appears to be orthogonal to the importance of recency effects—for example, the topic #fashion benefits from adding a foreground model, which once again seems intuitive (i.e., once a user mentions a particular item, it is more likely that other users will join in the conversation).

In the bottom block of Table 2, we fix a particular smoothing technique (Jelinek-Mercer, $\lambda = 0.4$) and history window size ($h = 10000$) and report the effects on perplexity for unigram language models using different history retention techniques (the queue, forget, and epoch methods described in Section 3.2). It is not surprising that the queue method outperforms the other two. The major weakness of the epoch approach is that it provides no mechanism to "forget" term frequencies over time, and therefore does not properly reward terms that appear recently. We are, however, somewhat surprised that the performance of the forget method is quite reasonable—for some topics, the difference in perplexity compared to the queue method is not very large. Given that the forget method requires far less space (we only need to keep track of the number of terms encountered, rather than the entire history of terms for the queue method), the higher perplexities might be a worthwhile tradeoff for some applications.

Table 3 shows perplexities of the bigram language models across all ten topics for different smoothing techniques, with the queue method ($h = 10000$). Once again, the smoothing parameters represent effective settings derived from a coarse exploration of the parameter space. For bigrams, experiments show that Jelinek-Mercer consistently has the lowest perplexity. It is no surprise to see large reductions in perplexity in moving from unigram to bigram language models. However, extrinsic evaluations show that lower perplexity does not translate into better task performance—as we shall see next. For space considerations, we omit results of additional experiments with bigram language models.

Topic	1	2	3	4	5	6	7	8	9	10
Background only	5851	5356	1676	11584	7257	3306	4442	10204	16887	7237
Queue ($h = 1000$)										
Absolute Discounting ($\delta = 0.9$)	**3754**	3578	1414	8902	4825	2798	4131	8318	9462	**3833**
Jelinek-Mercer ($\lambda = 0.3$)	3788	**3539**	**1408**	8817	**5009**	**2769**	3975	7994	**9212**	3919
Dirichlet ($\mu = 5000$)	3971	3699	1416	**8711**	5141	**2769**	**3866**	**7993**	9340	4148
Normalized Stupid Backoff ($\alpha = 1.0$)	4668	4474	1991	11865	6473	3947	5548	10225	11502	4697
Queue ($h = 10000$)										
Absolute Discounting ($\delta = 0.9$)	3696	3472	1382	8760	4538	2904	4099	8052	10595	3846
Jelinek-Mercer ($\lambda = 0.4$)	3652	3372	**1342**	**8329**	4582	**2721**	**3741**	**7595**	**9788**	3784
Dirichlet ($\mu = 10000$)	**3609**	**3336**	1353	8517	**4557**	2752	3813	7651	9969	**3738**
Normalized Stupid Backoff ($\alpha = 0.3$)	4526	4213	1778	12004	5893	3774	5311	10073	13605	4603
Jelinek-Mercer ($\lambda = 0.4, h = 10000$)										
Queue	**3652**	**3372**	**1342**	**8329**	**4582**	**2721**	**3741**	**7595**	**9788**	**3784**
Forget	3670	3421	1371	8575	4744	2762	3922	7849	9752	3790
Epoch	4772	4243	1426	9355	5428	2922	3852	7979	10800	4869

Table 2: Perplexity of unigram language models under different settings.

Topic	1	2	3	4	5	6	7	8	9	10
Background only	924	1505	150	2526	921	667	921	976	2179	1006
Queue ($h = 10000$)										
Absolute Discounting ($\delta = 0.8$)	451	573	97	1176	624	526	622	559	820	**318**
Jelinek-Mercer ($\lambda = 0.3$)	**365**	**438**	**95**	**1058**	**458**	**350**	**520**	**470**	**789**	356
Dirichlet ($\mu = 100$)	538	690	132	1518	687	562	713	674	1074	405
Normalized Stupid Backoff ($\alpha = 1.0$)	469	549	479	1366	568	431	673	638	1058	526

Table 3: Perplexity of bigram language models under different settings.

6. EXTRINSIC EVALUATION

In addition to intrinsic perplexity-based measures, we also evaluated the effectiveness of our topic language models on the task of topic tracking, conceived as a filtering problem on the tweet stream. We experimented with a simple perplexity-based classifier. For a given tweet, its perplexity with respect to the current model is computed. If the perplexity is above a certain decision threshold, we considered it relevant (i.e., on topic). By varying the decision threshold, we can realized different precision/recall tradeoffs.

A few more details about our experimental setup: To avoid the laborious task of building a test collection by hand, we focused our evaluation efforts on tweets that contained at least one hashtag. We processed all tweets with hashtags as follows: first, the ground truth label (i.e., topic of interest) is removed from the tweet. We then applied the perplexity-based classifier to obtain a classification decision. After recording the results, the foreground model is updated accordingly (if the tweet contains the ground truth label).

A few caveats on the evaluation: as with many other IR evaluations, absolute values of metrics can be misleading. For example, for the topic #nfl, there are other commonly-used tags such as #football that are roughly synonymous, or in many cases tweets are tagged with finer-grained entities such as team names. Our evaluation method would not be able to capture any of these natural variations. In many cases, it is nearly impossible to classify, even for a human, whether a tweet was topical or not. A tweet to the effect of "I'm watching #glee right now" (a popular TV show) provides little content-based signal as to the topic if the hashtag itself is removed. Finally, we perform no attempt at special processing of non-English tweets. Thus, when interpreting results, the relative effectiveness of different techniques is more important than the absolute numbers.

Figure 1 shows the precision-recall plots of our ten test topics using unigram LMs and the queue history retention approach ($h = 10000$). In each figure, we plot recall on the *x*-axis; since only a small fraction of tweets are topically relevant, the *y*-axis precision scale focuses on values close to one. In all plots, the dotted line shows the precision achievable by rejecting all tweets as non-relevant (a trivial baseline).

The striking conclusion is that although Jelinek-Mercer smoothing yields the lowest perplexities overall (from Table 2), in the extrinsic task normalized stupid backoff appears to be just as effective and in some cases more effective than the other smoothing techniques. This appears to be true for all ten topics with exception of topic 7 (#teaparty), where absolute discounting appears to perform the best. Normalized stupid backoff has the additional advantage in that it is the simplest, and hence fastest smoothing technique. We believe that the issue with Jelinek-Mercer smoothing is the λ setting, which gives relatively low weight to the foreground model—from a perplexity point of view, this allows the model to better account for background events, but this is detrimental to the extrinsic task, which requires more emphasis on recency.

Figure 2 compares the effects of different history sizes $h = \{1000, 10000\}$ with the queue method, showing absolute discounting and normalized stupid backoff for a single topic. The longer history outperforms the shorter history, as expected. We omit corresponding figures for other topics, since they are qualitatively similar and provide little additional insight.

Our final set of experiments compared unigram and bigram language models under the same conditions. Figure 3 shows results for the #nfl topic, comparing the background model and normalized stupid backoff (queue, $h = 1000$). Results from other topics look similar, and hence are omitted here. We see that unigram LMs are more effective than bigram LMs. It is perhaps not surprising,

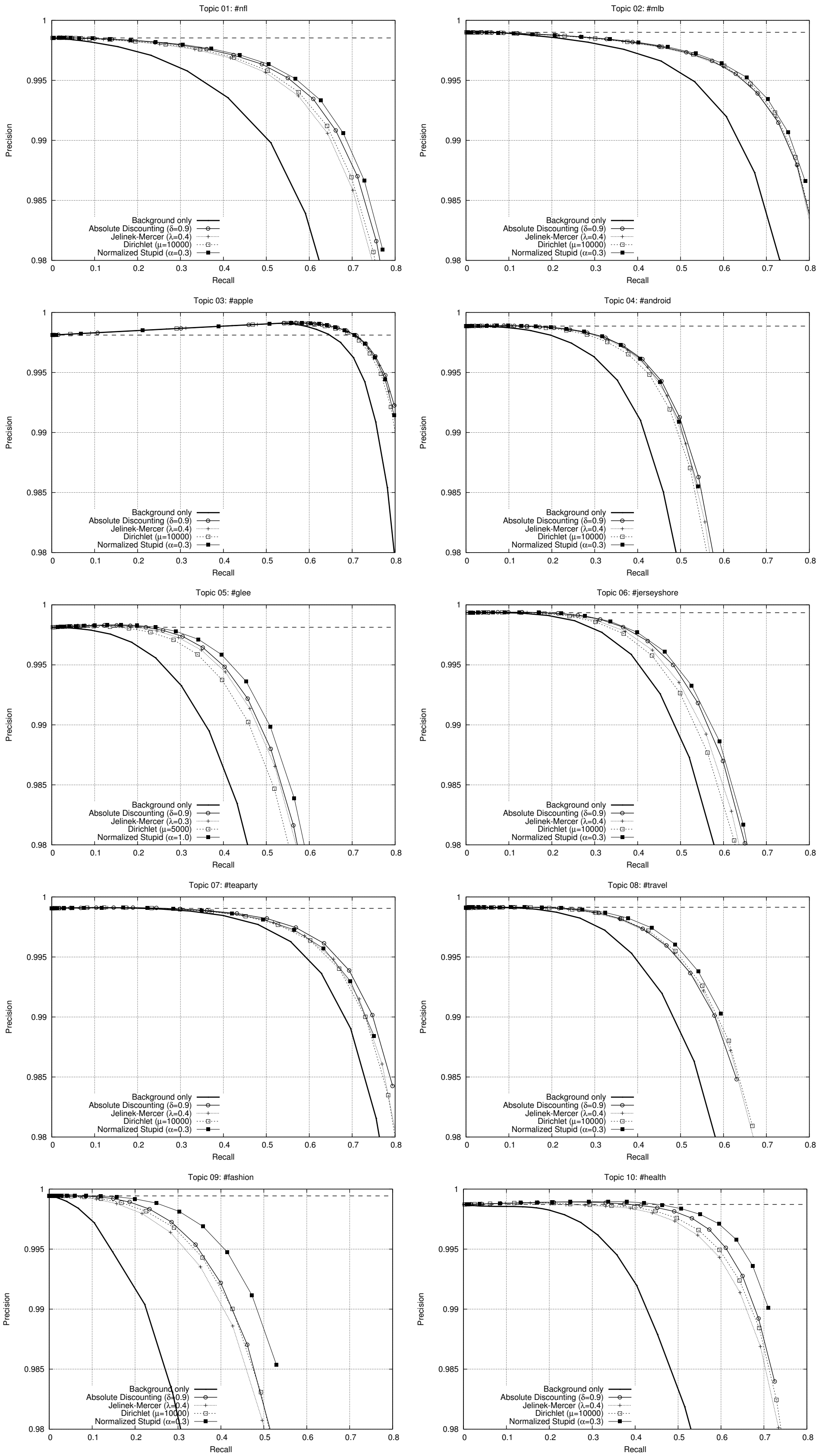

Figure 1: Precision-recall plots for the ten topics in our test set using unigram LMs and the queue method ($h = 10000$).

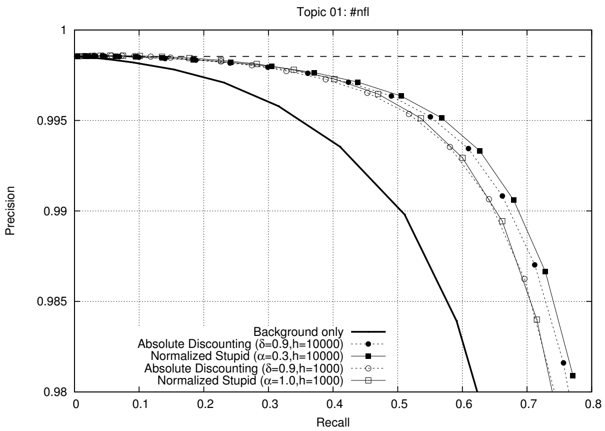

Figure 2: Comparison between different history sizes ($h = \{1000, 10000\}$), using the queue method, unigrams.

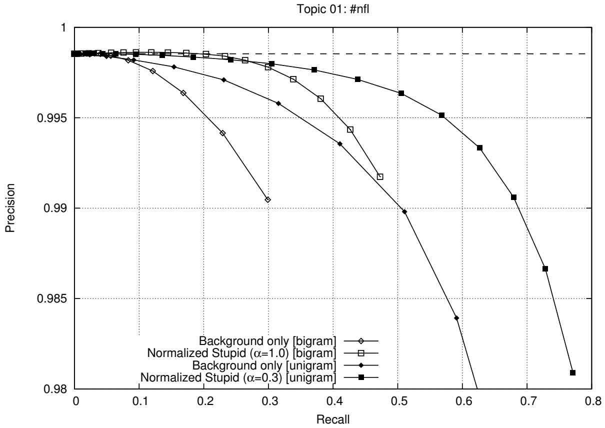

Figure 3: Comparison between unigram and bigram language models (background vs. stupid backoff).

as this conclusion is consistent with results from IR and other related tasks. Somewhat oversimplifying, topicality boils down to a problem of keyword spotting, for which unigram LMs are not only faster, but better. Based on examination of output, it appears that the bigram models are attempting to model aspects of tweet fluency (i.e., grammaticality), which is an issue completely orthogonal to topicality, the actual problem at hand. For this reason, with bigram LMs precision drops off rapidly with increasing recall: fluent tweets that are off topic are falsely classified as positive, while disfluent but topical tweets are rejected.

7. CONCLUSION

The problem of tracking topics in a continuous stream of short natural language texts, as applied to tweet streams, compels us to look for simple, efficient solutions. Our general approach is to combine a foreground model to capture "what's happening right now" with a background model to combat data sparsity. We found that stupid backoff, perhaps the simplest method to combine two probability distributions, is at least as effective as any of the more principled smoothing techniques we examined. A simple technique of retaining history in a FIFO queue was also found to be effective. The combination, applied to unigram language models, provides a good balance between maintaining recency and combating sparsity.

Beyond the immediate task, the results of this work provide more evidence that in "extreme" settings, simple approaches work well (in terms of balancing speed, effectiveness, scalability, and other real-world considerations). Brants et al. [5] demonstrated that at web-scale (i.e., extremely large datasets), stupid backoff outperforms Kneser-Ney smoothing. Here, we contribute findings along another dimension of "extreme": in rapidly evolving streams of natural language text (that, literally, change on a millisecond basis), the KISS principle remains a good starting point.

8. ACKNOWLEDGMENTS

We'd like to thank Abdur Chowdhury for helpful comments on earlier drafts of this paper. The first author is grateful to Esther and Kiri for their loving support and dedicates this work to Joshua and Jacob.

9. REFERENCES

[1] J. Allan. *Topic Detection and Tracking: Event-Based Information Organization*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[2] S. Asur and B. A. Huberman. Predicting the future with social media. Technical Report HPL-2010-53, HP Laboratories, 2010.

[3] S. Asur, B. A. Huberman, G. Szabo, and C. Wang. Trends in social media: Persistence and decay. Technical report, HP Laboratories, 2011.

[4] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts. Everyone's an influencer: Quantifying influence on Twitter. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining (WSDM 2011)*, pages 65–74, Hong Kong, China, 2011.

[5] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 858–867, Prague, Czech Republic, 2007.

[6] M. Cha, H. Haddadi, F. Benevenuto, and K. P. Gummadi. Measuring user influence in Twitter: The million follower fallacy. In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media (ICWSM 2010)*, pages 10–17, Washington, D.C., 2010.

[7] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL 1996)*, pages 310–318, Santa Cruz, California, 1996.

[8] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Çetintemel, Y. Xing, and S. B. Zdonik. Scalable distributed stream processing. In *Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, Asilomar, California, 2003.

[9] A. Goyal, H. Daumé, and S. Venkatasubramanian. Streaming for large scale NLP: Language modeling. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 512–520, Boulder, Colorado, 2009.

[10] D. J. Hopkins and G. King. A method of automated nonparametric content analysis for social science. *American Journal of Political Science*, 54(1):229–247, 2010.

[11] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Pearson, Upper Saddle River, New Jersey, 2009.

[12] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a social network or a news media? In *Proceedings of the 19th International World Wide Web Conference (WWW 2010)*, pages 591–600, Raleigh, North Carolina, 2010.

[13] K. Lerman and R. Ghosh. Information contagion: An empirical study of the spread of news on Digg and Twitter social networks. In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media (ICWSM 2010)*, pages 90–97, Washington, D.C., 2010.

[14] A. Levenberg and M. Osborne. Stream-based randomised language models for SMT. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*, pages 756–764, Singapore, 2009.

[15] X. Liu, S. Zhang, F. Wei, and M. Zhou. Recognizing named entities in tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2011)*, Portland, Oregon, 2011.

[16] S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Computer Science*, 1(2), 2005.

[17] B. O'Connor, R. Balasubramanyan, B. R. Routledge, and N. A. Smith. From Tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media (ICWSM 2010)*, pages 122–129, Washington, D.C., 2010.

[18] S. Petrović, M. Osborne, and V. Lavrenko. Streaming first story detection with application to Twitter. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL 2010)*, pages 181–189, Los Angeles, California, 2010.

[19] D. Ramage, S. Dumais, and D. Liebling. Characterizing microblogs with topic models. In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media (ICWSM 2010)*, pages 130–137, Washington, D.C., 2010.

[20] A. Ritter, C. Cherry, and B. Dolan. Unsupervised modeling of Twitter conversations. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL 2010)*, pages 172–180, Los Angeles, California, 2010.

[21] D. M. Romero, B. Meeder, and J. Kleinberg. Differences in the mechanics of information diffusion across topics: Idioms, political hashtags, and complex contagion on Twitter. In *Proceedings of the 20th International World Wide Web Conference (WWW 2011)*, pages 695–704, Hyderabad, India, 2011.

[22] J. Weng, E.-P. Lim, J. Jiang, and Q. He. TwitterRank: Finding topic-sensitive influential Twitterers. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM 2010)*, pages 261–270, New York, New York, 2010.