Dynamic Language Models for Streaming Text

Dani Yogatama* **Chong Wang*** **Bryan R. Routledge†** **Noah A. Smith*** **Eric P. Xing***
*School of Computer Science
†Tepper School of Business
Carnegie Mellon University
Pittsburgh, PA 15213, USA
*{dyogatama,chongw,nasmith,epxing}@cs.cmu.edu, †routledge@cmu.edu

Abstract

We present a probabilistic language model that captures temporal dynamics and conditions on arbitrary *non-linguistic* context features. These context features serve as important indicators of language changes that are otherwise difficult to capture using text data by itself. We learn our model in an efficient online fashion that is scalable for large, streaming data. With five streaming datasets from two different genres—economics news articles and social media—we evaluate our model on the task of sequential language modeling. Our model consistently outperforms competing models.

1 Introduction

Language models are a key component in many NLP applications, such as machine translation and exploratory corpus analysis. Language models are typically assumed to be static—the word-given-context distributions do not change over time. Examples include n-gram models (Jelinek, 1997) and probabilistic topic models like latent Dirichlet allocation (Blei et al., 2003); we use the term "language model" to refer broadly to probabilistic models of text.

Recently, streaming datasets (e.g., social media) have attracted much interest in NLP. Since such data evolve rapidly based on events in the real world, assuming a static language model becomes unrealistic. In general, more data is seen as better, but treating all past data equally runs the risk of distracting a model with irrelevant evidence. On the other hand, cautiously using only the most recent data risks overfitting to short-term trends and missing important time-insensitive effects (Blei and Lafferty, 2006; Wang et al., 2008). Therefore, in this paper, we take steps toward methods for capturing long-range temporal dynamics in language use.

Our model also exploits observable context variables to capture temporal variation that is otherwise difficult to capture using only text. Specifically for the applications we consider, we use stock market data as exogenous evidence on which the language model depends. For example, when an important company's price moves suddenly, the language model should be based not on the very recent history, but should be similar to the language model for a day when a similar change happened, since people are likely to say similar things (either about that company, or about conditions relevant to the change). Non-linguistic contexts such as stock price changes provide useful auxiliary information that might indicate the similarity of language models across different timesteps.

We also turn to a fully online learning framework (Cesa-Bianchi and Lugosi, 2006) to deal with non-stationarity and dynamics in the data that necessitate adaptation of the model to data in real time. In online learning, streaming examples are processed only when they arrive. Online learning also eliminates the need to store large amounts of data in memory. Strictly speaking, online learning is distinct from *stochastic* learning, which for language models built on massive datasets has been explored by Hoffman et al. (2013) and Wang et al. (2011). Those techniques are still for static modeling. Language modeling for streaming datasets in the context of machine translation was considered by Levenberg and Osborne (2009) and Levenberg et al. (2010). Goyal et al. (2009) introduced a streaming algorithm for large scale language modeling by approximating n-gram frequency counts. We propose a general online learning algorithm for language modeling that draws inspiration from regret minimization in sequential predictions (Cesa-Bianchi and Lugosi, 2006) and on-

Transactions of the Association for Computational Linguistics, 2 (2014) 181–192. Action Editor: Eric Fosler-Lussier.
Submitted 10/2013; Revised 2/2014; Published 4/2014.

line variational algorithms (Sato, 2001; Honkela and Valpola, 2003).

To our knowledge, our model is the first to bring together temporal dynamics, conditioning on non-linguistic context, and scalable online learning suitable for streaming data and extensible to include topics and n-gram histories. The main idea of our model is independent of the choice of the base language model (e.g., unigrams, bigrams, topic models, etc.). In this paper, we focus on unigram and bigram language models in order to evaluate the basic idea on well understood models, and to show how it can be extended to higher-order n-grams. We leave extensions to topic models for future work.

We propose a novel task to evaluate our proposed language model. The task is to predict economics-related text at a given time, taking into account the changes in stock prices up to the corresponding day. This can be seen an inverse of the setup considered by Lavrenko et al. (2000), where news is assumed to influence stock prices. We evaluate our model on economics news in various languages (English, German, and French), as well as Twitter data.

2 Background

In this section, we first discuss the background for sequential predictions then describe how to formulate online language modeling as sequential predictions.

2.1 Sequential Predictions

Let $w_1, w_2, \ldots, w_T$ be a sequence of response variables, revealed one at a time. The goal is to design a good learner to predict the next response, given previous responses and additional evidence which we denote by $\boldsymbol{x}_t \in \mathbb{R}^M$ (at time t). Throughout this paper, we use the term *features* for $\boldsymbol{x}$. Specifically, at each round t, the learner receives $\boldsymbol{x}_t$ and makes a prediction $\hat{w}_t$, by choosing a parameter vector $\boldsymbol{\alpha}_t \in \mathbb{R}^M$. In this paper, we refer to $\boldsymbol{\alpha}$ as *feature coefficients*.

There has been an enormous amount of work on online learning for sequential predictions, much of it building on convex optimization. For a sequence of loss functions $\ell_1, \ell_2, \ldots, \ell_T$ (parameterized by $\boldsymbol{\alpha}$), an online learning algorithm is a strategy to minimize the regret, with respect to the best fixed $\boldsymbol{\alpha}^*$ in hindsight.[1] Regret guarantees assume a Lipschitz condition on the loss function ℓ that can be prohibitive for complex models. See Cesa-Bianchi and Lugosi (2006), Rakhlin (2009), Bubeck (2011), and Shalev-Shwartz (2012) for in-depth discussion and review.

There has also been work on online and stochastic learning for Bayesian models (Sato, 2001; Honkela and Valpola, 2003; Hoffman et al., 2013), based on variational inference. The goal is to approximate posterior distributions of latent variables when examples arrive one at a time.

In this paper, we will use both kinds of techniques to learn language models for streaming datasets.

2.2 Problem Formulation

Consider an online language modeling problem, in the spirit of sequential predictions. The task is to build a language model that accurately predicts the texts generated on day t, conditioned on observable features up to day t, $\boldsymbol{x}_{1:t}$. Every day, after the model makes a prediction, the actual texts $\boldsymbol{w}_t$ are revealed and we suffer a loss. The loss is defined as the negative log likelihood of the model $\ell_t = -\log p(\boldsymbol{w}_t \mid \boldsymbol{\alpha}, \boldsymbol{\beta}_{1:t-1}, \boldsymbol{x}_{1:t-1}, \boldsymbol{n}_{1:t-1})$, where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}_{1:T}$ are the model parameters and $\boldsymbol{n}$ is a background distribution (details are given in §3.2). We can then update the model and proceed to day $t+1$. Notice the similarity to the sequential prediction described above. Importantly, this is a realistic setup for building evolving language models from large-scale streaming datasets.

3 Model

3.1 Notation

We index timesteps by $t \in \{1, \ldots, T\}$ and word types by $v \in \{1, \ldots, V\}$, both are always given as subscripts. We denote vectors in boldface and use $1:T$ as a shorthand for $\{1, 2, \ldots, T\}$. We assume words of the form $\{\boldsymbol{w}_t\}_{t=1}^T$ for $\boldsymbol{w}_t \in \mathbb{R}^V$, which is the vector of word frequences at timetstep t. Non-linguistic context features are $\{\boldsymbol{x}_t\}_{t=1}^T$ for $\boldsymbol{x}_t \in \mathbb{R}^M$. The goal is to learn parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}_{1:T}$, which will be described in detail next.

3.2 Generative Story

The main idea of our model is illustrated by the following generative story for the unigram language

[1] Formally, the regret is defined as $\text{Regret}_T(\boldsymbol{\alpha}^*) = \sum_{t=1}^T \ell_t(\boldsymbol{x}_t, \boldsymbol{\alpha}_t, w_t) - \inf_{\boldsymbol{\alpha}^*} \sum_{t=1}^T \ell_t(\boldsymbol{x}_t, \boldsymbol{\alpha}^*, w_t)$.

model. (We will discuss the extension to higher-order language models later.) A graphical representation of our proposed model is given in Figure 1.

1. Draw feature coefficients $\boldsymbol{\alpha} \sim \mathcal{N}(0, \lambda\mathbf{I})$.[2] Here $\boldsymbol{\alpha}$ is a vector in $\mathbb{R}^M$, where M is the dimensionality of the feature vector.
2. For each timestep t:
   (a) Observe non-linguistic context features $\boldsymbol{x}_t$.
   (b) Draw $\boldsymbol{\beta}_t \sim \mathcal{N}\left(\sum_{k=1}^{t-1} \delta_k \frac{\exp(\boldsymbol{\alpha}^\top \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{x}_k))}{\sum_{j=1}^{t-1} \delta_j \exp(\boldsymbol{\alpha}^\top \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{x}_j))} \boldsymbol{\beta}_k, \varphi\mathbf{I}\right)$. Here, $\boldsymbol{\beta}_t$ is a vector in $\mathbb{R}^V$, where V is the size of the word vocabulary, $\boldsymbol{\varphi}$ is the variance parameter and δ_k is a fixed hyperparameter; we discuss them below.
   (c) For each word $w_{t,v}$, draw $w_{t,v} \sim \text{Categorical}\left(\frac{\exp(n_{1:t-1,v} + \beta_{t,v})}{\sum_{j \in V} \exp(n_{1:t-1,j} + \beta_{t,j})}\right)$.

In the last step, $\boldsymbol{\beta}_t$ and $\boldsymbol{n}$ are mapped to the V-dimensional simplex, forming a distribution over words. $\boldsymbol{n}_{1:t-1} \in \mathbb{R}^V$ is a background (log) distribution, inspired by a similar idea in Eisenstein et al. (2011). In this paper, we set $n_{1:t-1,v}$ to be the log-frequency of v up to time $t-1$. We can interpret $\boldsymbol{\beta}$ as a time-dependent deviation from the background log-frequencies that incorporates world-context. This deviation comes in the form of a weighted average of earlier deviation vectors.

The intuition behind the model is that the probability of a word appearing at day t depends on the background log-frequencies, the deviation coefficients of the word at previous timesteps $\boldsymbol{\beta}_{1:t-1}$, and the similarity of current conditions of the world (based on observable features $\boldsymbol{x}$) to previous timesteps through $\boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{x}_k)$. That is, $\boldsymbol{f}$ is a function that takes d-dimensional feature vectors at two timesteps $\boldsymbol{x}_t$ and $\boldsymbol{x}_k$ and returns a similarity vector $\boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{x}_k) \in \mathbb{R}^M$ (see §6.1.1 for an example of $\boldsymbol{f}$ that we use in our experiments). The similarity is parameterized by $\boldsymbol{\alpha}$, and decays over time with rate δ_k. In this work, we assume a fixed window size c (i.e., we consider c most recent timesteps), so that $\delta_{1:t-c-1} = 0$ and $\delta_{t-c:t-1} = 1$. This allows up to cth order dependencies.[3] Setting δ this way allows us to bound the number of past vectors $\boldsymbol{\beta}$ that need to be kept in memory. We set $\boldsymbol{\beta}_0$ to $\mathbf{0}$.

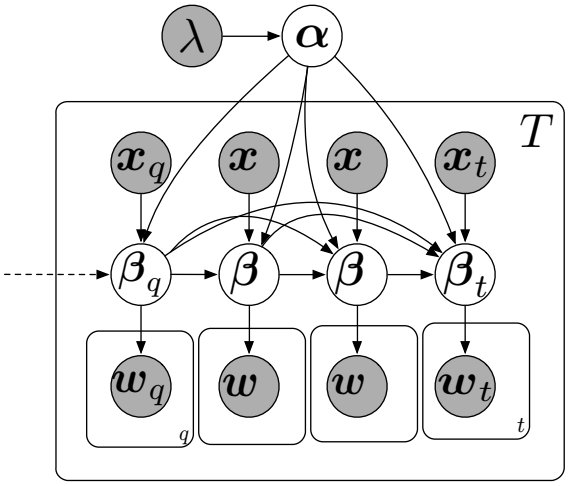

Figure 1: Graphical representation of the model. The subscript indices q, r, s are shorthands for the previous timesteps $t-3, t-2, t-1$. Only four timesteps are shown here. There are arrows from previous $\boldsymbol{\beta}_{t-4}, \boldsymbol{\beta}_{t-5}, \ldots, \boldsymbol{\beta}_{t-c}$ to $\boldsymbol{\beta}_t$, where c is the window size as described in §3.2. They are not shown here, for readability.

Although the generative story described above is for unigram language models, extensions can be made to more complex models (e.g., mixture of unigrams, topic models, etc.) and to longer n-gram contexts. In the case of topic models, the model will be related to dynamic topic models (Blei and Lafferty, 2006) augmented by context features, and the learning procedure in §4 can be used to perform online learning of dynamic topic models. However, our model captures longer-range dependencies than dynamic topic models, and can condition on non-linguistic features or metadata. In the case of higher-order n-grams, one simple way is to draw more $\boldsymbol{\beta}$, one for each history. For example, for a bigram model, $\boldsymbol{\beta}$ is in $\mathbb{R}^{V^2}$, rather than $\mathbb{R}^V$ in the unigram model. We consider both unigram and bigram language models in our experiments in §6. However, the main idea presented in this paper is largely independent of the base model.

Related work. Mimno and McCallum (2008) and Eisenstein et al. (2010) similarly conditioned text on

[2] Feature coefficients $\boldsymbol{\alpha}$ can be also drawn from other distributions such as $\boldsymbol{\alpha} \sim \text{Laplace}(0, \lambda)$.

[3] In online Bayesian learning, it is known that forgetting inaccurate estimates from earlier timesteps is important (Sato, 2001; Honkela and Valpola, 2003). Since we set $\delta_{1:t-c-1} = 0$, at every timestep t, δ_k leads to forgetting older examples.

observable features (e.g., author, publication venue, geography, and other document-level metadata), but conducted inference in a batch setting, thus their approaches are not suitable for streaming data. It is not immediately clear how to generalize their approach to dynamic settings. Algorithmically, our work comes closest to the online dynamic topic model of Iwata et al. (2010), except that we also incorporate context features.

4 Learning and Inference

The goal of the learning procedure is to minimize the overall negative log likelihood,

$$-\log \mathcal{L}(\mathcal{D}) = -\log \int d\boldsymbol{\beta}_{1:T} p(\boldsymbol{\beta}_{1:T} \mid \boldsymbol{\alpha}, \boldsymbol{x}_{1:T}) p(\boldsymbol{w}_{1:T} \mid \boldsymbol{\beta}_{1:T}, \boldsymbol{n}).$$

However, this quantity is intractable. Instead, we derive an upper bound for this quantity and minimize that upper bound. Using Jensen's inequality, the variational upper bound on the negative log likelihood is:

$$-\log \mathcal{L}(\mathcal{D}) \leq -\int d\boldsymbol{\beta}_{1:T} q(\boldsymbol{\beta}_{1:T} \mid \boldsymbol{\gamma}_{1:T}) \log \frac{p(\boldsymbol{\beta}_{1:T} \mid \boldsymbol{\alpha}, \boldsymbol{x}_{1:T}) p(\boldsymbol{w}_{1:T} \mid \boldsymbol{\beta}_{1:T}, \boldsymbol{n})}{q(\boldsymbol{\beta}_{1:T} \mid \boldsymbol{\gamma}_{1:T})}. \quad (4)$$

Specifically, we use mean-field variational inference where the variables in the variational distribution q are completely independent. We use Gaussian distributions as our variational distributions for $\boldsymbol{\beta}$, denoted by $\boldsymbol{\gamma}$ in the bound in Eq. 4. We denote the parameters of the Gaussian variational distribution for $\beta_{t,v}$ (word v at timestep t) by $\mu_{t,v}$ (mean) and $\sigma_{t,v}$ (variance).

Figure 2 shows the functional form of the variational bound that we seek to minimize, denoted by $\hat{\mathcal{B}}$. The two main steps in the optimization of the bound are inferring $\boldsymbol{\beta}_t$ and updating feature coefficients $\boldsymbol{\alpha}$. We next describe each step in detail.

4.1 Learning

The goal of the learning procedure is to minimize the upper bound in Figure 2 with respect to $\boldsymbol{\alpha}$. However, since the data arrives in an online fashion, and speed is very important for processing streaming datasets, the model needs to be updated at every timestep t (in our experiments, daily).

Notice that at timestep t, we only have access to $\boldsymbol{x}_{1:t}$ and $\boldsymbol{w}_{1:t}$, and we perform learning at *every* timestep *after* the text for the current timestep $\boldsymbol{w}_t$ is revealed. We do not know $\boldsymbol{x}_{t+1:T}$ and $\boldsymbol{w}_{t+1:T}$. Nonetheless, we want to update our model so that it can make a better prediction at $t+1$. Therefore, we can only minimize the bound until timestep t. Let $C_k \triangleq \frac{\exp(\boldsymbol{\alpha}^\top \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{x}_k))}{\sum_{j=t-c}^{t-1} \exp(\boldsymbol{\alpha}^\top \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{x}_j))}$. Our learning algorithm is a variational Expectation-Maximization algorithm (Wainwright and Jordan, 2008).

E-step Recall that we use variational inference and the variational parameters for $\boldsymbol{\beta}$ are $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. As shown in Figure 2, since the log-sum-exp in the last term of $\mathcal{B}$ is problematic, we introduce additional variational parameters $\boldsymbol{\zeta}$ to simplify $\mathcal{B}$ and obtain $\hat{\mathcal{B}}$ (Eqs. 2–3). The E-step deals with all the local variables $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, and $\boldsymbol{\zeta}$.

Fixing other variables and taking the derivative of the bound $\hat{\mathcal{B}}$ w.r.t. ζ_t and setting it to zero, we obtain the closed-form update for ζ_t: $\zeta_t = \sum_{v \in V} \exp(n_{1:t-1,v}) \exp\left(\mu_{t,v} + \frac{\sigma_{t,v}}{2}\right)$.

To minimize with respect to $\boldsymbol{\mu}_t$ and $\boldsymbol{\sigma}_t$, we apply gradient-based methods since there are no closed-form solutions. The derivative w.r.t. $\mu_{t,v}$ is:

$$\frac{\partial \hat{\mathcal{B}}}{\partial \mu_{t,v}} = \frac{\mu_{t,v} - C_k \mu_{k,v}}{\varphi} - n_{t,v} + \frac{n_t}{\zeta_t} \exp(n_{1:t-1,v}) \exp\left(\mu_{t,v} + \frac{\sigma_{t,v}}{2}\right),$$

where $n_t = \sum_{v \in V} n_{t,v}$.

The derivative w.r.t. $\sigma_{t,v}$ is:

$$\frac{\partial \hat{\mathcal{B}}}{\partial \sigma_{t,v}} = \frac{1}{2\sigma_{t,v}} + \frac{1}{2\varphi} + \frac{n_t}{2\zeta_t} \exp(n_{1:t-1,v}) \exp\left(\mu_{t,v} + \frac{\sigma_{t,v}}{2}\right).$$

Although we require iterative methods in the E-step, we find it to be reasonably fast in practice.[4] Specifically, we use the L-BFGS quasi-Newton algorithm (Liu and Nocedal, 1989).

We can further improve the bound by updating the variational parameters for timestep $1 : t-1$, i.e., $\boldsymbol{\mu}_{1:t-1}$ and $\boldsymbol{\sigma}_{1:t-1}$, as well. However, this will require storing the texts from previous timesteps. Additionally, this will complicate the M-step update described

[4] Approximately 16.5 seconds/day (walltime) to learn the model on the EN:NA dataset on a 2.40GHz CPU with 24GB memory.

$$\mathcal{B} = -\sum_{t=1}^{T} \mathbb{E}_q[\log p(\boldsymbol{\beta}_t \mid \boldsymbol{\beta}_k, \boldsymbol{\alpha}, \boldsymbol{x}_t)] - \sum_{t=1}^{T} \mathbb{E}_q[\log p(\boldsymbol{w}_t \mid \boldsymbol{\beta}_t, \boldsymbol{n}_t)] - \mathbb{H}(q) \tag{1}$$

$$= \sum_{t=1}^{T} \left\{ \frac{1}{2} \sum_{j \in V} \log \frac{\sigma_{t,j}}{\varphi} - \mathbb{E}_q \left[- \frac{\left(\boldsymbol{\beta}_t - \sum_{k=t-c}^{t-1} C_k \boldsymbol{\beta}_k\right)^2}{2\varphi} \right] - \mathbb{E}_q \left[\sum_{v \in \boldsymbol{w}_t} n_{1:t-1,v} + \beta_{t,v} - \log \sum_{j \in V} \exp(n_{1:t-1,j} + \beta_{t,j}) \right] \right\} \tag{2}$$

$$\leq \sum_{t=1}^{T} \left\{ \frac{1}{2} \sum_{j \in V} \log \frac{\sigma_{t,v}}{\varphi} + \frac{\left(\boldsymbol{\mu}_t - \sum_{k=t-c}^{t-1} C_k \boldsymbol{\mu}_k\right)^2}{2\varphi} + \frac{\boldsymbol{\sigma}_t + \sum_{k=t-c}^{t-1} C_k^2 \boldsymbol{\sigma}_k}{2\varphi} \right.$$

$$\left. - \sum_{v \in \boldsymbol{w}_t} \left(\mu_{t,v} - \log \zeta_t - \frac{1}{\zeta_t} \sum_{j \in V} \exp\left(n_{1:t-1,j}\right) \exp\left(\mu_{t,j} + \frac{\sigma_{t,j}}{2}\right) \right) \right\} + \text{const} \tag{3}$$

Figure 2: The variational bound that we seek to minimize, $\mathcal{B}$. $\mathbb{H}(q)$ is the entropy of the variational distribution q. The derivation from line 1 to line 2 is done by replacing the probability distributions $p(\boldsymbol{\beta}_t \mid \boldsymbol{\beta}_k, \boldsymbol{\alpha}, \boldsymbol{x}_t)$ and $p(\boldsymbol{w}_t \mid \boldsymbol{\beta}_t, \boldsymbol{n}_t)$ by their respective functional forms. Notice that in line 3 we compute the expectations under the variational distributions and further bound $\mathcal{B}$ by introducing additional variational parameters $\boldsymbol{\zeta}$ using Jensen's inequality on the log-sum-exp in the last term. We denote the new bound $\hat{\mathcal{B}}$.

below. Therefore, for each $s < t$, we choose to fix $\boldsymbol{\mu}_s$ and $\boldsymbol{\sigma}_s$ once they are learned at timestep s.

M-step In the M-step, we update the global parameter $\boldsymbol{\alpha}$, fixing $\boldsymbol{\mu}_{1:t}$. Fixing other parameters and taking the derivative of $\hat{\mathcal{B}}$ w.r.t. $\boldsymbol{\alpha}$, we obtain:[5]

$$\frac{\partial \hat{\mathcal{B}}}{\partial \alpha} = \frac{(\boldsymbol{\mu}_t - \sum_{k=t-c}^{t-1} C_k \boldsymbol{\mu}_k)(-\sum_{k=t-c}^{t-1} \frac{\partial C_k}{\partial \boldsymbol{\alpha}})}{\varphi} + \frac{\sum_{k=t-c}^{t-1} C_k \boldsymbol{\sigma}_k \frac{\partial C_k}{\partial \boldsymbol{\alpha}}}{\varphi},$$

where:

$$\frac{\partial C_k}{\partial \boldsymbol{\alpha}} = C_k \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{x}_k) - C_k \frac{\sum_{s=t-c}^{t-1} \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{x}_s) \exp(\boldsymbol{\alpha}^\top \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{x}_s))}{\sum_{s=t-c}^{t-1} \exp(\boldsymbol{\alpha}^\top \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{x}_s))}.$$

We follow the convex optimization strategy and simply perform a stochastic gradient update: $\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t + \eta_t \frac{\partial \hat{\mathcal{B}}}{\partial \boldsymbol{\alpha}_t}$ (Zinkevich, 2003). While the variational bound $\hat{\mathcal{B}}$ is not convex, given the local variables $\boldsymbol{\mu}_{1:t}$ and $\boldsymbol{\sigma}_{1:t}$, optimizing $\boldsymbol{\alpha}$ at timestep t without knowing the future becomes a convex problem.[6] Since we do not reestimate $\boldsymbol{\mu}_{1:t-1}$ and $\boldsymbol{\sigma}_{1:t-1}$ in the E-step, the choice to perform online gradient descent instead of iteratively performing batch optimization at every timestep is theoretically justified.

Notice that our overall learning procedure is still to minimize the variational upper bound $\hat{\mathcal{B}}$. All these choices are made to make the model suitable for learning in real time from large streaming datasets. Preliminary experiments showed that performing more than one EM iteration per day does not considerably improve performance, so in our experiments we perform one EM iteration per day.

To learn the parameters of the model, we rely on approximations and optimize an upper bound $\hat{\mathcal{B}}$. We have opted for this approach over alternatives (such as MCMC methods) because of our interest in the online, large-data setting. Our experiments show that we are still able to learn reasonable parameter estimates by optimizing $\hat{\mathcal{B}}$. Like online variational methods for other latent-variable models such as LDA (Sato, 2001; Hoffman et al., 2013), open questions remain about the tightness of such approximations and the identifiability of model parameters. We note, how-

[5] In our implementation, we augment $\boldsymbol{\alpha}$ with a squared L_2 regularization term (i.e., we assume that $\boldsymbol{\alpha}$ is drawn from a normal distribution with mean zero and variance λ) and use the FOBOS algorithm (Duchi and Singer, 2009). The derivative of the regularization term is simple and is not shown here. Of course, other regularizers (e.g., the L_1-norm, which we use for other parameters, or the $L_{1/\infty}$-norm) can also be explored.

[6] As a result, our algorithm is Hannan consistent w.r.t. the best fixed $\boldsymbol{\alpha}$ (for $\hat{\mathcal{B}}$) in hindsight; i.e., the average regret goes to zero as T goes to ∞.

ever, that our model does not include latent mixtures of topics and may be generally easier to estimate.

5 Prediction

As described in §2.2, our model is evaluated by the loss suffered at every timestep, where the loss is defined as the negative log likelihood of the model on text at timestep $\boldsymbol{w}_t$. Therefore, at each timestep t, we need to *predict* (the distribution of) $\boldsymbol{w}_t$. In order to do this, for each word $v \in V$, we simply compute the deviation means $\boldsymbol{\beta}_{t,v}$ as weighted combinations of previous means, where the weights are determined by the world-context similarity encoded in $\boldsymbol{x}$:

$$\mathbb{E}_q[\beta_{t,v} \mid \mu_{t,v}] = \sum_{k=t-c}^{t-1} \frac{\exp(\boldsymbol{\alpha}^\top \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{x}_k))}{\sum_{j=t-c}^{t-1} \exp(\boldsymbol{\alpha}^\top \boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{x}_j))} \mu_{k,v}.$$

Recall that the word distribution that we use for prediction is obtained by applying the operator π that maps $\boldsymbol{\beta}_t$ and $\boldsymbol{n}$ to the V-dimensional simplex, forming a distribution over words: $\pi(\boldsymbol{\beta}_t, \boldsymbol{n}_{1:t-1})_v = \frac{\exp(n_{1:t-1,v}+\beta_{t,v})}{\sum_{j\in V}\exp(n_{1:t-1,j}+\beta_{t,j})}$, where $n_{1:t-1,v} \in \mathbb{R}^V$ is a background distribution (the log-frequency of word v observed up to time $t-1$).

6 Experiments

In our experiments, we consider the problem of predicting economy-related text appearing in news and microblogs, based on observable features that reflect current economic conditions in the world at a given time. In the following, we describe our dataset in detail, then show experimental results on text prediction. In all experiments, we set the window size $c = 7$ (one week) or $c = 14$ (two weeks), $\lambda = \frac{1}{2|V|}$ (V is the size of vocabulary of the dataset under consideration), and $\varphi = 1$.

6.1 Dataset

Our data contains metadata and text corpora. The metadata is used as our features, whereas the text corpora are used for learning language models and predictions. The dataset (excluding Twitter) can be downloaded at `http://www.ark.cs.cmu.edu/DynamicLM`.

6.1.1 Metadata

We use end-of-day stock prices gathered from `finance.yahoo.com` for each stock included in the Standard & Poor's 500 index (S&P 500). The index includes large (by market value) companies listed on US stock exchanges.[7] We calculate daily (continuously compounded) returns for each stock, o: $r_{o,t} = \log P_{o,t} - \log P_{o,t-1}$, where $P_{o,t}$ is the closing stock price.[8] We make a simplifying assumption that text for day t is generated after $P_{o,t}$ is observed.[9] In general, stocks trade Monday to Friday (except for federal holidays and natural disasters). For days when stocks do not trade, we set $r_{o,t} = 0$ for all stocks since any price change is not observed.

We transform returns into similarity values as follows: $f(x_{o,t}, x_{o,k}) = 1$ iff $\text{sign}(r_{o,t}) = \text{sign}(r_{o,k})$ and 0 otherwise. While this limits the model by ignoring the magnitude of price changes, it is still reasonable to capture the similarity between two days.[10] There are 500 stocks in the S&P 500, so $\boldsymbol{x}_t \in \mathbb{R}^{500}$ and $\boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{x}_k) \in \mathbb{R}^{500}$.

6.1.2 Text data

We have five streams of text data. The first four corpora are news streams tracked through Reuters.[11] Two of them are written in English, North American Business Report (EN:NA) and Japanese Investment News (EN:JP). The remaining two are German Economic News Service (DE, in German) and French Economic News Service (FR, in French). For all four of the Reuters streams, we collected news data over a period of thirteen months (392 days), 2012-05-26 to 2013-06-21. See Table 1 for descriptive statistics of these datasets. Numerical terms are mapped to a single word, and all letters are downcased.

The last text stream comes from the Decahose/Gardenhose stream from Twitter. We collected public tweets that contain ticker symbols (i.e., symbols that are used to denote stocks of a particular company in a stock market), preceded by the dollar

[7] For a list of companies listed in the S&P 500 as of 2012, see `http://en.wikipedia.org/wiki/List_of_S\%26P_500_companies`. This set was fixed during the time periods of all our experiments.

[8] We use the "adjusted close" on Yahoo that includes interim dividend cash flows and also adjusts for "splits" (changes in the number of outstanding shares).

[9] This is done in order to avoid having to deal with hourly timesteps. In addition, intraday price data is only available through commercial data provided.

[10] Note that daily stock returns are equally likely to be positive or negative and display little serial correlation.

[11] `http://www.reuters.com`

Dataset	Total # Doc.	Avg. # Doc.	#Days	Unigrams		Bigrams	
				Total # Tokens	Size Vocab.	Total # Tokens	Size Vocab.
EN:NA	86,683	223	392	28,265,550	10,000	11,804,201	5,000
EN:JP	70.807	182	392	16,026,380	10,000	7,047,095	5,000
FR	62,355	160	392	11,942,271	10,000	3,773,517	5,000
DE	51,515	132	392	9,027,823	10,000	3,499,965	5,000
Twitter	214,794	336	639	1,660,874	10,000	551,768	5,000

Table 1: Statistics about the datasets. Average number of documents (third column) is per day.

sign $ (e.g., $GOOG, $MSFT, $AAPL, etc.). These tags are generally used to indicate tweets about the stock market. We look at tweets from the period 2011-01-01 to 2012-09-30 (639 days). As a result, we have approximately 100–800 tweets per day. We tokenized the tweets using the CMU ARK TweetNLP tools,[12] numerical terms are mapped to a single word, and all letters are downcased.

We perform two experiments using unigram and bigram language models as the base models. For each dataset, we consider the top 10,000 unigrams after removing corpus-specific stopwords (the top 100 words with highest frequencies). For the bigram experiments, we only use 5,000 words to limit the number of unique bigrams so that we can simulate experiments for the entire time horizon in a reasonable amount of time. In standard open-vocabulary language modeling experiments, the treatment of unknown words deserves care. We have opted for a controlled, closed-vocabulary experiment, since standard smoothing techniques will almost surely interact with temporal dynamics and context in interesting ways that are out of scope in the present work.

6.2 Baselines

Since this is a forecasting task, at each timestep, we only have access to data from previous timesteps. Our model assumes that all words in all documents in a corpus come from a single multinomial distribution. Therefore, we compare our approach to the corresponding base models (standard unigram and bigram language models) over the same vocabulary (for each stream). The first one maintains counts of every word and updates the counts at each timestep. This corresponds to a base model that uses all of the available data up to the current timestep ("base all"). The second one *replaces* counts of every word with the counts from the previous timestep ("base one"). Additionally, we also compare with a base model whose counts decay exponentially ("base exp"). That is, the counts from previous timesteps decay by $\exp(-\gamma s)$, where s is the distance between previous timesteps and the current timestep and γ is the decay constant. We set the decay constant $\gamma = 1$. We put a symmetric Dirichlet prior on the counts ("add-one" smoothing); this is analogous to our treatment of the background frequencies $\boldsymbol{n}$ in our model. Note that our model, similar to "base all," uses *all* available data up to timestep $t-1$ when making predictions for timestep t. The window size c only determines which previous timesteps' models can be chosen for making a prediction today. The past models themselves are estimated from all available data up to their respective timesteps.

We also compare with two strong baselines: a linear interpolation of "base one" models for the past week ("int. week") and a linear interpolation of "base all" and "base one" ("int one all"). The interpolation weights are learned online using the normalized exponentiated gradient algorithm (Kivinen and Warmuth, 1997), which has been shown to enjoy a stronger regret guarantee compared to standard online gradient descent for learning a convex combination of weights.

6.3 Results

We evaluate the perplexity on unseen dataset to evaluate the performance of our model. Specifically, we use per-word predictive perplexity:

$$\text{perplexity} = \exp\left(-\frac{\sum_{t=1}^{T} \log p(\boldsymbol{w}_t \mid \boldsymbol{\alpha}, \boldsymbol{x}_{1:t}, \boldsymbol{n}_{1:t-1})}{\sum_{t=1}^{T}\sum_{j\in V} w_{t,j}}\right).$$

Note that the denominator is the number of tokens up to timestep T. Lower perplexity is better.

Table 2 and Table 3 show the perplexity results for

[12] https://www.ark.cs.cmu.edu/TweetNLP

Dataset	base all	base one	base exp	int. week	int. one all	$c = 7$	$c = 14$
EN:NA	3,341	3,677	3,486	3,403	3,271	**3,262**	3,285
EN:JP	2,802	3,212	2,750	2,949	2,708	**2,656**	2,689
FR	3,603	3,910	3,678	3,625	3,416	**3,404**	3,438
DE	3,789	4,199	3,979	3,926	**3,634**	3,649	3,687
Twitter	3,880	6,168	5,133	5,859	4,047	**3,801**	3,819

Table 2: Perplexity results for our five data streams in the unigram experiments. The base models in "base all," "base one," and "base exp" are unigram language models. "int. week" is a linear interpolation of "base one" from the past week. "int. one all" is a linear interpolation of "base one" and "base all". The rightmost two columns are versions of our model. Best results are highlighted in bold.

Dataset	base all	base one	base exp	int. week	int. one all	$c = 7$
EN:NA	242	2,229	1,880	2,200	244	**223**
EN:JP	185	2,101	1,726	2,050	189	**167**
FR	159	2,084	1,707	2,068	166	**139**
DE	268	2,634	2,267	2,644	282	**243**
Twitter	756	4,245	4,253	5,859	4,046	**739**

Table 3: Perplexity results for our five data streams in the bigram experiments. The base models in "base all," "base one," and "base exp" are bigram language models. "int. week" is a linear interpolation of "base one" from the past week. "int. one all" is a linear interpolation of "base one" and "base all". The rightmost column is a version of our model with $c = 7$. Best results are highlighted in bold.

each of the datasets for unigram and bigram experiments respectively. Our model outperformed other competing models in all cases but one. Recall that we only define the similarity function of world context as: $f(x_{o,t}, x_{o,k}) = 1$ iff $\text{sign}(r_{o,t}) = \text{sign}(r_{o,k})$ and 0 otherwise. A better similarity function (e.g., one that takes into account market size of the company and the magnitude of increase or decrease in the stock price) might be able to improve the performance further. We leave this for future work. Furthermore, the variations can be captured using models from the past week. We discuss why increasing c from 7 to 14 did not improve performance of the model in more detail in §6.4.

We can also see how the models performed over time. Figure 4 traces perplexity for four Reuters news stream datasets.[13] We can see that in some cases the performance of the "base all" model degraded over time, whereas our model is more robust to temporal shifts.

In the bigram experiments, we only ran our model with $c = 7$, since we need to maintain $\boldsymbol{\beta}$ in $\mathbb{R}^{V^2}$, instead of $\mathbb{R}^V$ in the unigram model. The goal of this experiment is to determine whether our method still adds benefit to more expressive language models. Note that the weights of the linear interpolation models are also learned in an online fashion since there are no classical training, development, and test sets in our setting. Since the "base one" model performed poorly in this experiment, the performance of the interpolated models also suffered. For example, the "int. one all" model needed time to learn that the "base one" model has to be downweighted (we started with all interpolated models having uniform weights), so it was not able to outperform even the "base all" model.

6.4 Analysis and Discussion

It should not be surprising that conditioning on world-context reduces perplexity (Cover and Thomas, 1991). A key attraction of our model, we believe, lies in the ability to inspect its parameters.

Deviation coefficients. Inspecting the model allows us to gain insight into temporal trends. We

[13] In both experiments, in order to manage the time and space complexities of updating $\boldsymbol{\beta}$, we apply a sparsity shrinkage technique by using OWL-QN (Andrew and Gao, 2007) when maximizing it, with regularization constant set to 1. Intuitively, this is equivalent to encouraging the deviation vector to be sparse (Eisenstein et al., 2011).

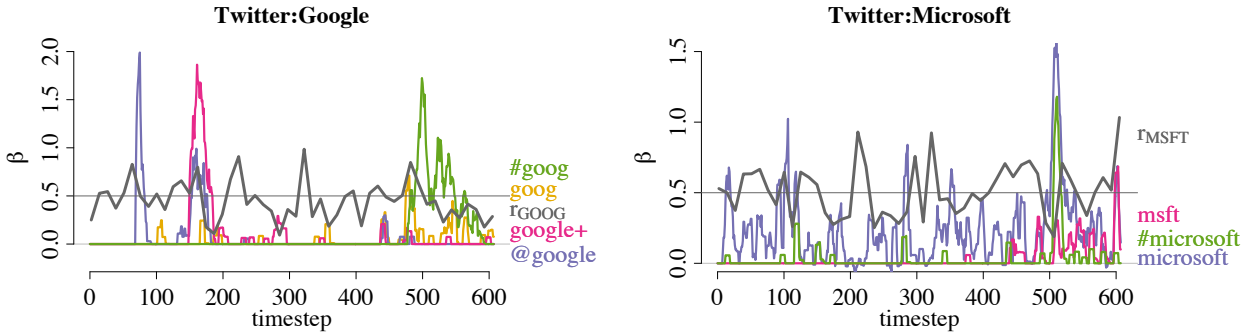

Figure 3: Deviation coefficients $\boldsymbol{\beta}$ over time for Google- and Microsoft-related words on Twitter with unigram base model ($c = 7$). Significant changes (increases or decreases) in the returns of Google and Microsoft stocks are usually followed by increases in β of related words.

investigate the deviations learned by our model on the Twitter dataset. Examples are shown in Figure 3. The left plot shows β for four words related to Google: `goog`, `#goog`, `@google`, `google+`. For comparison, we also show the return of Google stock for the corresponding timestep (scaled by 50 and centered at 0.5 for readability, smoothed using loess (Cleveland, 1979), denoted by r_{GOOG} in the plot). We can see that significant changes of return of Google stocks (e.g., the r_{GOOG} spikes between timesteps 50–100, 150–200, 490–550 in the plot) occurred alongside an increase in β of Google-related words. Similar trends can also be observed for Microsoft-related words in the right plot. The most significant loss of return of Microsoft stocks (the downward spike near timestep 500 in the plot) is followed by a sudden sharp increase in β of the words `#microsoft` and `microsoft`.

Feature coefficients. We can also inspect the learned feature coefficients $\boldsymbol{\alpha}$ to investigate which stocks have higher associations with the text that is generated. Our feature coefficients are designed to reflect which changes (or lack of changes) in stock prices influence the word distribution more, not which stocks are talked about more often. We find that the feature coefficients do not correlate with obvious company characteristics like market capitalization (firm size). For example, on the Twitter dataset with bigram base models, the five stocks with the highest weights are: ConAgra Foods Inc., Intel Corp., Bristol-Myers Squibb, Frontier Communications Corp., and Amazon.com Inc. Strongly negative weights tended to align with streams with less activity, suggesting that these were being used to smooth across all c days of history. A higher weight for stock o implies an increase in probability of choosing models from previous timesteps s, when the state of the world for the current timestep t and timestep s is the same (as represented by our similarity function) with respect to stock o (all other things being equal), and a decrease in probability for a lower weight.

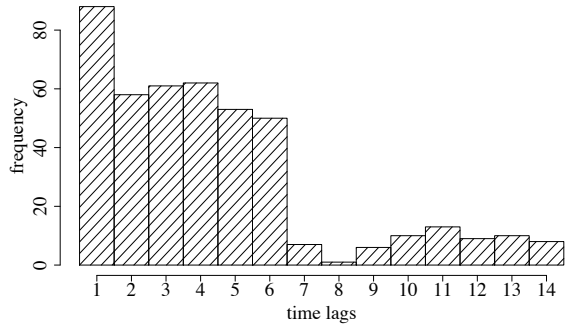

Figure 5: Distributions of the selection probabilities of models from the previous $c = 14$ timesteps, on the EN:NA dataset with unigram base model. For simplicity, we show E-step modes. The histogram shows that the model tends to favor models from days closer to the current date.

Selected models. Besides feature coefficients, our model captures temporal shift by modeling similarity across the most recent c days. During inference, our model weights different word distributions from the past. The similarity is encoded in the pairwise features $\boldsymbol{f}(\boldsymbol{x}_t, \boldsymbol{x}_k)$ and the parameters $\boldsymbol{\alpha}$. Figure 5 shows the distributions of the strongest-posterior models from previous timesteps, based on how far

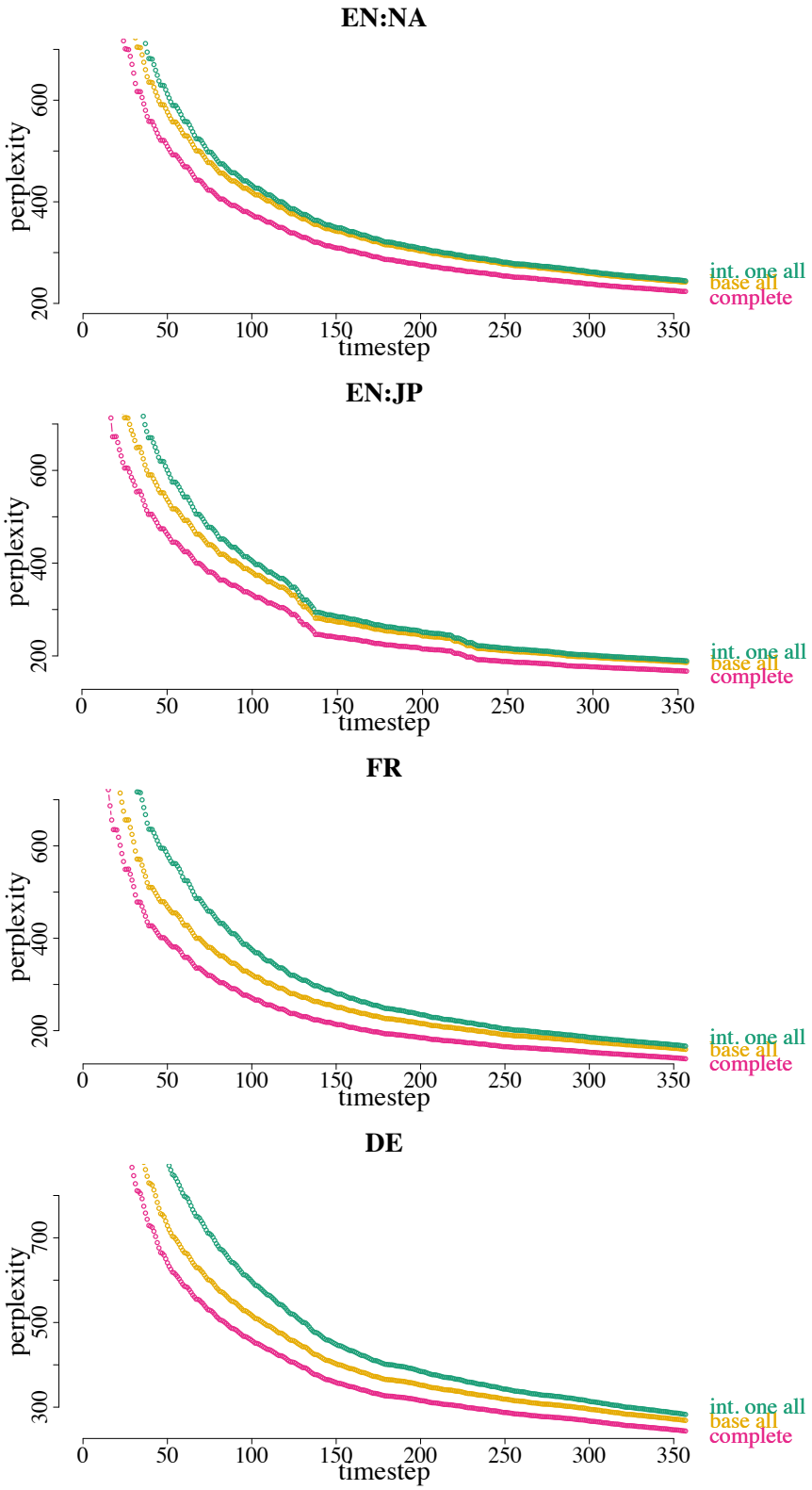

Figure 4: Perplexity over time for four Reuters news streams ($c = 7$) with bigram base models.

in the past they are at the time of use, aggregated across rounds on the EN:NA dataset, for window size $c = 14$. It shows that the model tends to favor models from days closer to the current date, with the $t - 1$ models selected the most, perhaps because the state of the world today is more similar to dates closer to today compare to more distant dates. The plot also explains why increasing c from 7 to 14 did not improve performance of the model, since most of the variation in our datasets can be captured with models from the past week.

Topics. Latent topic variables have often figured heavily in approaches to dynamic language modeling. In preliminary experiments incorporating single-membership topic variables (i.e., each document belongs to a single topic, as in a mixture of unigrams), we saw no benefit to perplexity. Incorporating topics also increases computational cost, since we must maintain and estimate one language model per topic, per timestep. It is straightforward to design models that incorporate topics with single- or mixed-membership as in LDA (Blei et al., 2003), an interesting future direction.

Potential applications. Dynamic language models like ours can be potentially useful in many applications, either as a standalone language model, e.g., predictive text input, whose performance may depend on the temporal dimension; or as a component in applications like machine translation or speech recognition. Additionally, the model can be seen as a step towards enhancing text understanding with numerical, contextual data.

7 Conclusion

We presented a dynamic language model for streaming datasets that allows conditioning on observable real-world context variables, exemplified in our experiments by stock market data. We showed how to perform learning and inference in an online fashion for this model. Our experiments showed the predictive benefit of such conditioning and online learning by comparing to similar models that ignore temporal dimensions and observable variables that influence the text.

Acknowledgements

The authors thank several anonymous reviewers for helpful feedback on earlier drafts of this paper and Brendan O'Connor for help with collecting Twitter data. This research was supported in part by Google, by computing resources at the Pittsburgh Supercomputing Center, by National Science Foundation grant IIS-1111142, AFOSR grant FA95501010247, ONR grant N000140910758, and by the Intelligence Advanced Research Projects Activity via Department of Interior National Business Center contract number D12PC00347. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

References

Galen Andrew and Jianfeng Gao. 2007. Scalable training of l_1-regularized log-linear models. In *Proc. of ICML*.

David M. Blei and John D. Lafferty. 2006. Dynamic topic models. In *Proc. of ICML*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Sébastien Bubeck. 2011. Introduction to online optimization. Technical report, Department of Operations Research and Financial Engineering, Princeton University.

Nicolò Cesa-Bianchi and Gábor Lugosi. 2006. *Prediction, Learning, and Games*. Cambridge University Press.

William S. Cleveland. 1979. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368):829–836.

Thomas M. Cover and Joy A. Thomas. 1991. *Elements of Information Theory*. John Wiley & Sons.

John Duchi and Yoram Singer. 2009. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10(7):2899–2934.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A latent variable model for geographic lexical variation. In *Proc. of EMNLP*.

Jacob Eisenstein, Amr Ahmed, and Eric P. Xing. 2011. Sparse additive generative models of text. In *Proc. of ICML*.

Amit Goyal, Hal Daume III, and Suresh Venkatasubramanian. 2009. Streaming for large scale NLP: Language modeling. In *Proc. of HLT-NAACL*.

Matt Hoffman, David M. Blei, Chong Wang, and John Paisley. 2013. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347.

Antti Honkela and Harri Valpola. 2003. On-line variational Bayesian learning. In *Proc. of ICA*.

Tomoharu Iwata, Takeshi Yamada, Yasushi Sakurai, and Naonori Ueda. 2010. Online multiscale dynamic topic models. In *Proc. of KDD*.

Frederick Jelinek. 1997. *Statistical Methods for Speech Recognition*. MIT Press.

Jyrki Kivinen and Manfred K. Warmuth. 1997. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132:1–63.

Victor Lavrenko, Matt Schmill, Dawn Lawrie, Paul Ogilvie, David Jensen, and James Allan. 2000. Mining of concurrent text and time series. In *Proc. of KDD Workshop on Text Mining*.

Abby Levenberg and Miles Osborne. 2009. Stream-based randomised language models for SMT. In *Proc. of EMNLP*.

Abby Levenberg, Chris Callison-Burch, and Miles Osborne. 2010. Stream-based translation models for statistical machine translation. In *Proc. of HLT-NAACL*.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming B*, 45(3):503–528.

David Mimno and Andrew McCallum. 2008. Topic models conditioned on arbitrary features with Dirichlet-multinomial regression. In *Proc. of UAI*.

Alexander Rakhlin. 2009. Lecture notes on online learning. Technical report, Department of Statistics, The Wharton School, University of Pennsylvania.

Masaaki Sato. 2001. Online model selection based on the variational bayes. *Neural Computation*, 13(7):1649–1681.

Shai Shalev-Shwartz. 2012. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194.

Martin J. Wainwright and Michael I. Jordan. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305.

Chong Wang, David M. Blei, and David Heckerman. 2008. Continuous time dynamic topic models. In *Proc. of UAI*.

Chong Wang, John Paisley, and David M. Blei. 2011. Online variational inference for the hierarchical Dirichlet process. In *Proc. of AISTATS*.

Martin Zinkevich. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *Proc. of ICML*.