Emerging Topic Detection using Dictionary Learning

Shiva Prasad Kasiviswanathan
IBM T.J. Watson Research
kasivisw@gmail.com

Prem Melville
IBM T.J. Watson Research
pmelvil@us.ibm.com

Arindam Banerjee
University of Minnesota
banerjee@cs.umn.edu

Vikas Sindhwani
IBM T.J. Watson Research
vsindhw@us.ibm.com

ABSTRACT

Streaming user-generated content in the form of blogs, microblogs, forums, and multimedia sharing sites, provides a rich source of data from which invaluable information and insights maybe gleaned. Given the vast volume of such social media data being continually generated, one of the challenges is to automatically tease apart the emerging topics of discussion from the constant background chatter. Such emerging topics can be identified by the appearance of multiple posts on a unique subject matter, which is distinct from previous online discourse. We address the problem of identifying emerging topics through the use of dictionary learning. We propose a two stage approach respectively based on detection and clustering of novel user-generated content. We derive a scalable approach by using the alternating directions method to solve the resulting optimization problems. Empirical results show that our proposed approach is more effective than several baselines in detecting emerging topics in traditional news story and newsgroup data. We also demonstrate the practical application to social media analysis, based on a study on streaming data from Twitter.

1. INTRODUCTION

The wide-spread popularity of social media, such as blogs and Twitter, has made it the focal point of online discussion and breaking news. Given the speed at which such user-generated content is produced, news flashes often occur on social media before they appear in traditional media outlets. Twitter, in particular, has been at the forefront of updates on disasters, such as earthquakes, on the 2009 post-election protests in Iran, and even on news of celebrity deaths [29]. Identifying such trending topics is of great interest beyond just reporting news, with applications to marketing, disease control, national security and many more. The business case for marketing and PR is particularly compelling, given that 19% of all tweets [16] and 32% of blog posts [26] talk about products or brands. Businesses need to be aware of what consumers, in general, are saying about their products; especially since any emerging negative information or opinions are best dealt with promptly.

Motivated by this need, we focus on the task of automatically detecting emerging topics, hot topics, or buzz from streams of documents/posts. For a subject to be considered an emerging topic, it must have *support*, in that it must appear from multiple sources; and it must be *novel*, in that it should be different from topics that have been, or are already, popular and well-known. Several current techniques that can be applied to address this problem, do not adequately meet both these criteria.

One possible approach to identifying emerging topics in document streams is to take all new posts and identify sets of posts that are similar. Extensive research in the area of document clustering and topic modeling with Latent Dirichlet Allocation [5], Probabilistic Latent Semantic Analysis [15], and Non-negative Matrix Factorizations [20] can be brought to bear here. Although clustering and topic modeling techniques (including, the dynamic ones) can find sets of posts expressing cohesive patterns of discussion, they are not guaranteed to identify clusters that are also novel or informative compared to previously appearing topics.

An alternative approach is to use techniques that have been developed for First Story Detection (FSD) [2], in the context of Topic Detection and Tracking (TDT) for traditional news streams. While seemingly related, FSD is only focused on detecting when a document discusses a previously unseen event. While first story detection by itself is very valuable for broadcast news, given the low signal-to-noise ratio in social media, it is less effective. In the space of social media, many posts that may be considered as "first stories" solely because they are very different from previous posts, may actually be of little value, e.g., the fact that *intlevg* on Twitter is eating "honey roasted peanuts for lunch", may be of little interest to the population at large.

For successful emerging topic detection, we need to identify several recent posts that are both similar to each other, and are dissimilar to previous posts. In this paper, we propose an approach based on sparse coding, in which data vectors are modeled as sparse linear combinations of basis elements. A stream of documents (where each document is modeled a vector $\mathbf{x} \in \mathbb{R}^m$) can be used to learn a *dictionary* ($A \in \mathbb{R}^{m \times k}$) of k *atoms*, such that the documents can be *approximately* represented by a linear combination of a *few* atoms. If a new document cannot be represented with low error as a sparse linear combination of these atoms, it is a good indicator of novelty of the document. Novel document, thus identified, are used to learn a new dictionary of novel topics. This new dictionary is then used to cluster similar novel posts together, which we identify as the emerging topic clusters.

We validated our approach on several datasets from broadcast news, news groups, and Twitter. Empirical results show that our approach is more effective at detecting emerging topics in terms of precision and recall, than an approach based on first story detec-

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.

tion followed by clustering of the first stories detected, and other variations. This paper makes the following main contributions:

(1) We formulate the task of detecting novel signals in streaming datasets as a sparse signal representation problem. A signal is represented with a sparse code over an existing dictionary along with a sparse error term. A novel signal is detected based on the lack of sparsity in such a representation. While our main application is emerging topic detection on streaming text, our methodology applies more broadly to other domains.

(2) Our objective function is a combination of the ℓ_1-norms of a sparse error (robust reconstruction) and a sparse code, i.e., $\|\mathbf{e}\|_1 + \lambda\|\mathbf{x}\|_1$ (for the signal $\mathbf{y} = A\mathbf{x}+\mathbf{e}$), which appears well suited for sparse high-dimensional datasets such as those that arise in text applications. Additionally, we have non-negativity constraints on the sparse code ($\mathbf{x}$) and dictionary (A), to maintain interpretability. Such a natural formulation combining sparsity, robustness, and non-negativity appears novel and highly appropriate for the task.

(3) Most algorithms for dictionary learning are iterative batch procedures, that operate on the entire training set, which is not feasible for very large and dynamic data. We present an online algorithm which computes the sparse coding for each data point only once, instead of an iterative batch update scheme. As a result, our approach is very scalable and well-suited for streaming data, such as from social media.

(4) We use a new practical alternating direction method (ADM) to solve various optimization problems appearing in our formulation, that may be of independent interest. ADM has recently gathered significant attention in the Machine Learning community due to its wide applicability to a range of learning problems with complex objective functions [6].

(5) We introduce a new dictionary learning based text clustering algorithm which, in our setting, outperforms the widely used Spherical K-Means [12] algorithm for clustering novel documents. The new algorithm is more generally applicable, and inherits the scalability of dictionary learning methods.

Notation. We use $[n]$ to denote the set $\{1, \dots, n\}$. Vectors are always column vectors and are denoted by boldface letters. For a scalar $r \in \mathbb{R}$, let $\text{sign}(r)$ denote the sign of r and $\text{soft}(r, T) = \text{sign}(r) \cdot \max\{|r| - T, 0\}$. The operator soft is extended to a matrix by applying it to every entry in the matrix. For arbitrary real matrices the standard inner product is defined as $\langle A, B\rangle = Tr(A^\top B)$, and the (squared) *Frobenius* matrix norm $\|A\|_{\text{fro}}^2 = \langle A, A\rangle$ is the sum of all squared entries of the matrix. We use s.t. as an abbreviation for subject to.

2. BACKGROUND WORK

Sparse error recovery has found success in applications such as image and speech processing [7], computer vision and pattern recognition [23]. One of most successful (and relevant to this paper) applications is in robust face recognition [36, 35]. It is known that a well-aligned frontal face image under different lighting and expression lies close to a special low-dimensional linear subspace spanned by the training samples from the same subject [4]. This observation has led to face recognition to be cast as a *sparse representation* problem where the objective is to recover a high-dimensional sparse signal $\mathbf{x} \in \mathbb{R}^k$ from a highly compressed linear measurement $\mathbf{y} = A\mathbf{x} + \mathbf{e} \in \mathbb{R}^m$ (where $\mathbf{e}$ is an unknown error vector) [35]. In recent work, Wright *et al*. [36] proposed to estimate $\mathbf{w} = [\mathbf{x}, \mathbf{e}]$ jointly as the sparsest solution to the extended equation: $\min_{\mathbf{w}} \|\mathbf{w}\|_1$ s.t. $\mathbf{y} = [A, \mathbb{I}]\mathbf{w}$, where $\mathbb{I}$ is the identity matrix. They show this formulation performs exceedingly well for robust face recognition. For example, they show that this ℓ_1-minimization enables almost perfect recognition even if more than 60% pixels of the query image are severely corrupted. In a later paper, Wright and Ma [35] provided theoretical justification for this ℓ_1-minimization formulation. We use a similar formulation to construct a sparse representation (see Section 4). One crucial difference between our approach and that advocated by Wright *et al*. is that our matrix A does not contain all the documents, but is a *compact* representative dictionary of the documents seen before. This makes our approach more scalable.

Dictionary learning falls into a general category of techniques known as *matrix factorization*. In this paper, we additionally enforce non-negativity constraint on the factor matrices and such factorizations have been widely studied as *non-negative matrix factorization* (NMF) [20]. Most NMF formulations deal with squared loss [20, 22] . Several papers have considered ℓ_1-regularized matrix factorizations with applications to topic models in text, e.g., [17, 39] but with an ℓ_2-loss function. Similar to our loss function (6), Ke and Kanade [19] consider an ℓ_1-loss function for non-negative matrix factorization, but do not impose sparsity in the factors. In their paper, Ke and Kanade [19] gave a simple alternative convex programming approach for solving non-negative matrix factorization with ℓ_1-loss. Another framework with closely related goals is that of Robust PCA [8] where a decomposition of a matrix is sought in terms of a low-rank component with small nuclear norm and sparse errors. However, the repeated need to solve linear/quadratic programs or SVD in these approaches not scalable to large matrices. Our use of the alternating directions method allows us to scale-up to large matrices that are typically encountered in the text domain. For a review of ADMs and their suitability to a variety of large scale learning problems, we point the reader to [6]. To the best of our knowledge, our approach is the first to propose a scalable algorithm for ℓ_1-regularized dictionary learning with robust ℓ_1-reconstruction error with an application in temporal topic analysis of streaming documents.

3. TASK DEFINITION

In order to represent documents in our data stream, we will use the conventional vector space model with TF-IDF (Term Frequency-Inverse Document Frequency) term-weighting [24]. Additionally, each document has a timestamp that indicates when the document arrives. The timestamp could be at the granularity of our liking, e.g., it could be the day or the exact time the document arrives. As new documents come in and new terms are identified, our vocabulary set increases, but for simplicity, we are going to work with a global vocabulary space containing m terms that is independent of t. The extension to the case where the vocabulary set increases with t is quite simple, and just requires adjusting the size of the matrices by zero-padding.

Let $\{P_t \in \mathbb{R}^{m \times n_t},\ t = 1, 2, \dots\}$ denote a sequence of streaming matrices, where P_t represents the sparse term-document matrix whose unit ℓ_1-normalized columns are the documents with timestamp t and n_t is the number of documents with timestamp t. Analogous to P_t, we also define $P_{\leq t}$ which is constructed by using all documents whose timestamp is $\leq t$. Let $n_{\leq t}$ be the number of documents with timestamp $\leq t$, then $P_{\leq t} \in \mathbb{R}^{m \times n_{\leq t}}$. Informally, the goal of *emerging topic detection* is to identify sets of documents in P_t that are similar to each other and are dissimilar to documents in $P_{\leq t-1}$.

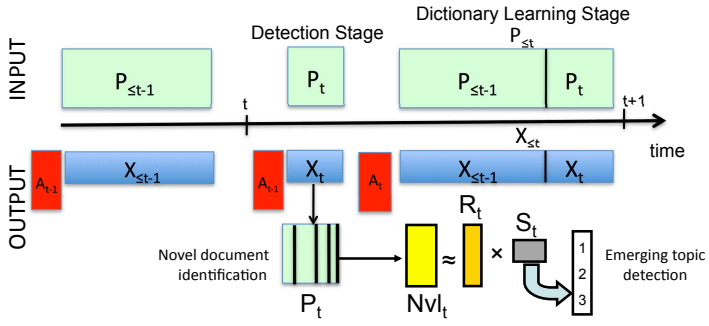

Figure 1: Schematic of our framework: At each timestep, we solve a sparse coding/dictionary learning problem with ℓ_1 reconstruction error and ℓ_1 regularization. The reconstruction error is used to identify novel documents which are again clustered using our framework to report emerging topics.

4. OUR FORMULATION

We split the task of emerging topic detection into two sub-tasks: (1) Identifying the novel documents (Nvl_t) from P_t (novel documents are those that belong to any of the emerging topics at time t), and (2) Clustering the novel documents. We use dictionary learning to solve both these sub-tasks (see Figure 1).

4.1 Identifying Novel Documents

Let $A_{t-1} \in \mathbb{R}^{m \times k}$ represent the dictionary matrix after time $t-1$; where dictionary A_{t-1} is a compact summary representation of all the documents in $P_{\leq t-1}$. Each column of A_{t-1} is called a *basis vector* or *atom*. The exact construction of the dictionary is described later, but ideally we want the dictionary to have a set of representative atoms for each of the old topics. With such a representative dictionary, documents from old topics can be represented as a linear combination of the atoms corresponding to that topic.

Given a new document vector $\mathbf{y}$ with timestamp t, we see if $\mathbf{y}$ could be represented as a sparse linear combination of the columns of A_{t-1}. The sparsest representation is the solution of

$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \text{ s.t. } \mathbf{y} = A_{t-1}\mathbf{x}, \mathbf{x} \geq 0, \tag{1}$$

where $\|\cdot\|_0$ is the ℓ_0-norm, counting the non-zero entries of a vector. However, in the general case, solving (1) is NP-hard and also hard to approximate [3]. Recently, a series of papers (see [9, 13] and references therein) have shown that under some favorable conditions one could obtain the solution to (1) by solving the following

$$\min_{\mathbf{x}} \|\mathbf{x}\|_1 \text{ s.t. } \mathbf{y} = A_{t-1}\mathbf{x}, \mathbf{x} \geq 0. \tag{2}$$

In essence, (2) can be viewed as a convex relaxation of (1). We will work with the ℓ_1-norm based convex relaxation in the sequel.

In most practical situations, (2) is not applicable because it may not be possible to represent $\mathbf{y}$ as $A_{t-1}\mathbf{x}$, e.g., if $\mathbf{y}$ has new words which are absent (i.e., have no support) in A_{t-1}. In such cases, one could represent $\mathbf{y} = A_{t-1}\mathbf{x} + \mathbf{e}$ where $\mathbf{e}$ is an unknown noise vector. Normally, if one considers a specific topic over time, typically a small set of new words get introduced in the discussion, e.g., if the topic is a sporting event like the Olympics, with time the discussion shifts to different athletes or events. The error vector $\mathbf{e}$ captures these terms. Since there are few such new words, the vector $\mathbf{e}$ is sparse. However, these terms do get used with much higher frequency within few documents than one might expect from the overall statistics of the corpus. So even though $\mathbf{e}$ is sparse it has some large, impulsive values. A natural relaxation to (2) for handling noise is $\min_{\mathbf{x}} \|\mathbf{x}\|_1$ s.t. $\|\mathbf{y} - A_{t-1}\mathbf{x}\|_2 \leq \delta, \mathbf{x} \geq 0$. The above problem (generally without the non-negativity constraint) is commonly known as the *constraint basis pursuit denoising problem* [10] and its variant

$$\min_{\mathbf{x}} \|\mathbf{y} - A_{t-1}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1 \text{ s.t. } \mathbf{x} \geq 0, \tag{3}$$

is referred to as the unconstrained basis pursuit denoising problem, also known as the Lasso [31]. Here, $\delta, \lambda > 0$ are parameters. The formulation (3) naturally takes into account both the reconstruction error (with the $\|\mathbf{y} - A_{t-1}\mathbf{x}\|_2^2$ term) and the complexity of the sparse decomposition (with the $\|\mathbf{x}\|_1$ term).

In the presence of isotopic Gaussian noise the ℓ_2-penalty on $\mathbf{e} = \mathbf{y} - A_{t-1}\mathbf{x}$ gives the best approximation of $\mathbf{x}$ [36, 37, 38]. However, for text documents (and in most other real scenarios), the noise vector $\mathbf{e}$ rarely satisfies the Gaussian assumption, and some of its coefficients contain large, impulsive values. In such scenarios, the ℓ_2-penalty of (3) may give an extremely bad approximation of $\mathbf{x}$ [37, 27]. However, in such real-world scenarios recent results [19, 36, 38, 35] have shown that imposing an ℓ_1-reconstruction error gives a more robust and better approximation of $\mathbf{x}$. We refer readers to these papers to get a more detailed exposition of the advantages of ℓ_1 over ℓ_2-reconstruction error. The basic intuition is that the ability of ℓ_1-penalty to recover the true solution $\mathbf{x}$ is independent of the magnitude of the $\mathbf{e}$, and depends only on the signs of $\mathbf{e}$ and the relative geometry of the column space of A and the unit ℓ_1 ball [36].

We use the following ℓ_1-formulation to recover $\mathbf{x}$

$$\min_{\mathbf{x}} \|\mathbf{y} - A_{t-1}\mathbf{x}\|_1 + \lambda\|\mathbf{x}\|_1 \text{ s.t. } \mathbf{x} \geq 0. \tag{4}$$

It is well-known that unlike (3) where squared ℓ_2-norm is used on the error, the ℓ_1-reconstruction error makes (4) an exact penalty method in the sense that it reduces to (2) when $\lambda > 0$ is less than some threshold [38]. Additionally, Yang *et al*. [38] claim that even without impulsive noise the ℓ_1-reconstruction error does not harm the solution quality as long as the data does not contain a large amount of Gaussian noise.

Given a new document $\mathbf{y}$ with timestamp of t and a dictionary A_{t-1}, we solve (4) to determine whether $\mathbf{y}$ is *novel* (with respect to dictionary A_{t-1}) or not. If the objective value of (4) is "small," then $\mathbf{y}$ is well-reconstructed by a linear combination of some basis vectors in A_{t-1}. We mark such documents as *non-novel* and discard them. Now, if the objective value is "large," then $\mathbf{y}$ has no good reconstruction among the basis vectors of the previous topics, thus suggesting novelty of $\mathbf{y}$. We add such documents to the set Nvl_t. Note that all documents are normalized to unit ℓ_1 length, and hence the objective values are in the same scale. The performance of this algorithm depends on the "quality" of the dictionary. We now describe our dictionary learning formulation.

4.2 Dictionary Learning

For simplicity of explanation, we assume that the dictionary is of fixed size $\mathbb{R}^{m \times k}$ (independent of t). Classic dictionary learning techniques [28, 1, 21] consider a finite training set of signals $S = [\mathbf{s}_1, \ldots, \mathbf{s}_n] \in \mathbb{R}^{m \times n}$ and optimize the empirical cost function $\sum_{i=1}^{n} l(\mathbf{s}_i, A)$, where $A \in \mathbb{R}^{m \times k}$ is the dictionary and $l(\cdot, \cdot)$ is a loss function such that $l(\mathbf{s}, A)$ should be small if A is "good" at representing the signal $\mathbf{s}$ in a sparse fashion. The value k is referred to as the size of the dictionary. A typical choice of the loss function is the Lasso-style objective

$$l(\mathbf{s}, A) = \min_{\mathbf{x}} \|\mathbf{s} - A\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1. \tag{5}$$

However, as we mentioned above in the absence of Gaussian noise using ℓ_1-loss on the error improves the estimation, therefore, we use the following loss function $l(\mathbf{s}, A) = \min_{\mathbf{x}} \|\mathbf{s} - A\mathbf{x}\|_1 + \lambda\|\mathbf{x}\|_1$. Additionally, we want dictionary A and $\mathbf{x}$'s to contain non-negative entries. Therefore, the problem of dictionary learning becomes

$$\min_{\mathbf{x}_i, A} \sum_{i=1}^{n} \|\mathbf{s}_i - A\mathbf{x}_i\|_1 + \lambda\|\mathbf{x}_i\|_1 \text{ s.t. } A \geq 0, \mathbf{x}_i \geq 0 \, \forall i \in [n].$$

In our setting at time t, we want to update the dictionary so that it forms a compact summary representation of all the documents in $P_{\leq t}$. Let $P_{\leq t} = [\mathbf{p}_1, \ldots, \mathbf{p}_{n_{\leq t}}] \in \mathbb{R}^{m \times n_{\leq t}}$. We learn a dictionary for time t by minimizing

$$\min_{\mathbf{x}_i, A} \sum_{i=1}^{n_{\leq t}} \|\mathbf{p}_i - A\mathbf{x}_i\|_1 + \lambda\|\mathbf{x}_i\|_1 \text{ s.t. } A \geq 0, \mathbf{x}_i \geq 0 \; \forall i \in [n_{\leq t}].$$

Equivalently this could be written as minimizing the following function over (X, A):

$$f(X, A) = \|P_{\leq t} - AX\|_1 + \lambda\|X\|_1 \text{ s.t. } A, X \geq 0. \quad (6)$$

Since we are factorizing $P_{\leq t}$ into two non-negative matrices, Equation (6) (without the $\lambda\|X\|_1$ term) is referred to as ℓ_1-loss non-negative factorization problem [20]. The optimization problem (6) is in general non-convex. But if one of the variables, either A or X is known, the objective function with respect to the other variable becomes a convex function (in fact, a linear function) and the global solution to (6) can be found. This iterative alternative minimization is the core idea behind most algorithms for dictionary learning [28, 1, 21]. However, these algorithms access the whole of the dataset in each iteration, and therefore, these algorithms cannot efficiently deal with large datasets. To overcome this problem, we use an online version of dictionary learning, where we update only A and use X obtained from previous stages of the algorithm (see Section 5).

4.3 Novel Documents to Emerging Topics

We now describe our procedure for going from novel documents to emerging topics. The idea is to again use dictionary learning. Given as input a set of (novel) documents and the number of topics (k_1) to be generated, we use a suitable modification of (6) to do detect emerging topics. The idea is as follows: If Nvl_t represents the set of novel documents, we learn a dictionary with k_1 atoms, where each atom corresponds to an emerging topic. In other words, we minimize the following function over (R, S):

$$f(R, S) = \|\mathrm{Nvl}_t - RS\|_1 + \lambda\|S\|_1 \text{ s.t. } R, S \geq 0, \quad (7)$$

where $R \in \mathbb{R}^{m \times k_1}$ is the dictionary matrix of term-topic association and $S \in \mathbb{R}^{k_1 \times |\mathrm{Nvl}_t|}$ is the matrix of topic-document association. The distribution of terms in each column of R is the latent representation of the k_1 emerging topics in the data. In order to present these topics in a meaningful way to users, we represent each topic by the most relevant documents to the topic. We do this by assigning each document to the atom in which it has the most dominant representation as given by the matrix S. This gives a clustering of novel documents into emerging topics.

5. OUR ALGORITHM

We present in this section our algorithm for detecting emerging topics. Our main procedure is summarized in Algorithm NVLCLUST. The algorithm alternates between a "detection stage" and a "dictionary learning stage." The detection stage at time t gets as input the dictionary A_{t-1} and P_t, and for each document $\mathbf{p}_j$ in P_t computes the best representation of $\mathbf{p}_j$ in terms of A_{t-1} by solving (4) (where $\mathbf{y}$ is replaced by $\mathbf{p}_j$). It is quite easy to transform (4) into a linear program. However, in our experiments the linear programming approach turned out to be quite slow (even when using commercial solvers like CPLEX). To speedup our algorithm, we use the alternating directions method (also known in the literature as alternating directions method of multipliers) [11, 38]. We explain this approach in Section 6. We classify a document $\mathbf{p}_j$ as novel if the objective value of (4) is above some chosen threshold ζ. Let $\mathrm{Nvl}_t \subseteq P_t$ be the set of document that are marked as novel at time t. The set of novel documents is then passed as input to Algorithm DICTCLUST which does clustering of these documents as explained in Section 4.3. Since, the size of Nvl_t is typically small, we solve it using a simple iterative batch procedure, alternatively fixing R, S and updating the other using the method of alternating directions.

We perform the dictionary learning stage in an online fashion. Online dictionary learning was recently introduced by Mairal *et al*. [22] who showed that it provides a scalable approach for handling large dynamic datasets. They consider an ℓ_2-loss function and show that their online algorithm converges to the minimum objective value in the stochastic case. Our online dictionary learning framework has similar structure to that of Mairal *et al*., although we focus on ℓ_1-loss. In the online setting, instead of (6), we update the dictionary by minimizing the following function over A:

$$f_X(A) = \|P_{\leq t} - AX_{\leq t}\|_1 + \lambda\|X_{\leq t}\|_1 \text{ s.t. } A \geq 0, \quad (8)$$

where $X_{\leq t} = [\mathbf{x}_1, \ldots, \mathbf{x}_{n_{\leq t}}]$ are computed during the previous detection stages. Notice that $\min_{A \geq 0} f_X(A)$ is an upper bound on $\min_{A, X \geq 0} f(X, A)$ (as we only optimize over A in $f_X(A)$). However, unlike $f(X, A)$ (which is not convex), minimizing $f_X(A)$ is a convex program (in fact, it can be re-cast as a linear program). For efficiency purposes, we use the method of alternating directions to compute $f_X(A)$ (see Section 6). The matrix A that minimizes the right hand side of (8) is the new dictionary A_t.

Remark. Our framework allows for several variations. By default, the optimization of A is initialized using A_{t-1} and therefore the number of topics tracked by the system is static. Alternatively, the set of emerging topics as discovered by the procedure described in Section 4.3 may also be explicitly injected as columns (in addition to A_{t-1}) in the initialization of A. In this case, the number of topics tracked steadily grows. A guided process may also be possible where a user scans the list of emerging topics and selectively introduces topics of interest in the dictionary. For simplicity, in this paper we use the default variation.

6. ALTERNATING DIRECTIONS METHOD

To speedup the algorithms NVLCLUST and DICTCLUST, we use the method of alternating directions to solve the various optimization problems. The reader is referred to [38, 6] and references therein for details and historical development of the alternating direction method (ADM). We start with a brief review of the general framework of ADM from [38]. Let $p(\mathbf{x}) : \mathbb{R}^a \to \mathbb{R}$ and $q(\mathbf{y}) : \mathbb{R}^b \to \mathbb{R}$ be convex functions, $F \in \mathbb{R}^{c \times a}$, $G \in \mathbb{R}^{c \times b}$, and $\mathbf{z} \in \mathbb{R}^c$. Consider the following optimization problem

$$\min_{\mathbf{x}, \mathbf{y}} p(\mathbf{x}) + q(\mathbf{y}) \text{ s.t. } F\mathbf{x} + G\mathbf{y} = \mathbf{z}, \quad (9)$$

where the variable vectors $\mathbf{x}$ and $\mathbf{y}$ are separate in the objective, and coupled only in the constraint. The augmented Lagrangian for

ADM EQUATIONS FOR SOLVING (10)

Input: $A_{t-1} \in \mathbb{R}^{m\times k}, \mathbf{p}_j \in \mathbb{R}^m, \lambda, \beta, \tau, \gamma > 0$

1. $\mathbf{x}_{(1)} \leftarrow 0, \mathbf{e}_{(1)} \leftarrow 0, \rho_{(1)} \leftarrow 0$
2. **for** $i = 1, 2, \ldots$, *convergence* **do**
   a) $\mathbf{e}_{(i+1)} \leftarrow \text{soft}(\mathbf{p}_j - A_{t-1}\mathbf{x}_{(i)} + \rho_{(i)}/\beta, 1/\beta)$
   b) $\mathbf{g}_{(i)} \leftarrow A_{t-1}^\top(A_{t-1}\mathbf{x}_{(i)} + \mathbf{e}_{(i+1)} - \mathbf{p}_j - \rho_{(i)}/\beta)$
   c) $\mathbf{x}_{(i+1)} \leftarrow \max\{\mathbf{x}_{(i)} - \tau\mathbf{g}_{(i)} - (\lambda\tau)/\beta, 0\}$
   d) $\rho_{(i+1)} \leftarrow \rho_{(i)} + \gamma\beta(\mathbf{p}_j - A_{t-1}\mathbf{x}_{(i+1)} - \mathbf{e}_{(i+1)})$

ADM EQUATIONS FOR SOLVING (8)

Input: $A_{t-1} \in \mathbb{R}^{m\times k}, P_{\le t} \in \mathbb{R}^{m\times n_{\le t}}, X_{\le t} \in \mathbb{R}^{m\times n_{\le t}}, \beta, \tau, \gamma > 0$

1. $A_{(1)} \leftarrow A_{t-1}, \Gamma_{(1)} \leftarrow P_{\le t} - A_{t-1}X_{\le t}, \Delta_{(1)} \leftarrow 0$
2. **for** $i = 1, 2, \ldots$, *convergence* **do**
   a) $\Gamma_{(i+1)} \leftarrow \text{soft}(P_{\le t} - A_{(i)}X_{\le t} + \Delta_{(i)}/\beta, 1/\beta)$
   b) $G_{(i)} \leftarrow (A_{(i)}X_{\le t} + \Gamma_{(i+1)} - P_{\le t} - \Delta_{(i)}/\beta)X_{\le t}^\top$
   c) $A_{(i+1)} \leftarrow \max\{A_{(i)} - \tau G_{(i)}, 0\}$
   d) $\Delta_{(i+1)} \leftarrow \Delta_{(i)} + \gamma\beta(P_{\le t} - A_{(i+1)}X_{\le t} - \Gamma_{(i+1)})$

Figure 2: ADM equations for detection and dictionary learning stages. The operator soft along with other notations are defined at the end of the Section 1.

ALGORITHM NVLCLUST: NOVEL TOPIC CLUSTERING

Input: $P_t = [\mathbf{p}_1, \ldots, \mathbf{p}_{n_t}] \in \mathbb{R}^{m\times n_t}, A_{t-1} \in \mathbb{R}^{m\times k}, \lambda, \zeta \in \mathbb{R}$
Detection stage:

1. $\text{Nvl}_t \leftarrow \emptyset$
2. **for each** $\mathbf{p}_j \in P_t$ **do**
   a) $\mathbf{x}_j = \text{argmin}_{\mathbf{x}\ge 0}\|\mathbf{p}_j - A_{t-1}\mathbf{x}\|_1 + \lambda\|\mathbf{x}\|_1$
   b) **if** $\|\mathbf{p}_j - A_{t-1}\mathbf{x}_j\|_1 + \lambda\|\mathbf{x}_j\|_1 > \zeta$ **then** $\text{Nvl}_t = \text{Nvl}_t \cup \{\mathbf{p}_j\}$
3. Invoke Algorithm DICTCLUST on Nvl_t

Dictionary Learning stage:

4. $X_{\le t} \leftarrow [X_{\le t-1} \,|\, \mathbf{x}_1, \ldots, \mathbf{x}_{n_t}]$
5. $P_{\le t} \leftarrow [P_{\le t-1} \,|\, \mathbf{p}_1, \ldots, \mathbf{p}_{n_t}]$
6. $A_t = \text{argmin}_{A\ge 0}\|P_{\le t} - AX_{\le t}\|_1 + \lambda\|X_{\le t}\|_1$

ALGORITHM DICTCLUST: DICTIONARY-BASED CLUSTERING

Input: Matrix of documents $\text{Nvl}_t, \lambda \in \mathbb{R}, k_1 \in \mathbb{R}$

1. $R_t, S_t = \text{argmin}_{R\ge 0, S\ge 0}\|\text{Nvl}_t - RS\|_1 + \lambda\|S\|_1$
2. let $S_t = [\mathbf{s}_1, \ldots, \mathbf{s}_{|\text{Nvl}_t|}]$ where each $\mathbf{s}_j \in \mathbb{R}^{k_1}$
3. **for each** document $\mathbf{p}_j \in \text{Nvl}_t$ **do**
   a) $l = \text{argmax}_j\{\mathbf{s}_j\}$
   b) assign document $\mathbf{p}_j$ to topic l

the above problem is given by

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \rho) = p(\mathbf{x}) + q(\mathbf{y}) + \rho^\top(\mathbf{z} - F\mathbf{x} - G\mathbf{y}) + \frac{\beta}{2}\|\mathbf{z} - F\mathbf{x} - G\mathbf{y}\|_2^2 ,$$

where $\rho \in \mathbb{R}^c$ is the Lagrangian multiplier and $\beta > 0$ is a penalty parameter.

ADM utilizes the separability form of (9) and replaces the joint minimization over $\mathbf{x}$ and $\mathbf{y}$ with two simpler problems. The ADM first minimizes $\mathcal{L}$ over $\mathbf{x}$, then over $\mathbf{y}$, and then applies a proximal minimization step with respect to the Lagrange multiplier ρ. In the ith iteration of the ADM procedure, given $(\mathbf{x}_{(i)}, \mathbf{y}_{(i)}, \rho_{(i)})$[1], we obtain $(\mathbf{x}_{(i+1)}, \mathbf{y}_{(i+1)}, \rho_{(i+1)})$ as follows

$$\begin{cases} \mathbf{x}_{(i+1)} \leftarrow \text{argmin}_{\mathbf{x}}\mathcal{L}(\mathbf{x}, \mathbf{y}_{(i)}, \rho_{(i)}), \\ \mathbf{y}_{(i+1)} \leftarrow \text{argmin}_{\mathbf{y}}\mathcal{L}(\mathbf{x}_{(i+1)}, \mathbf{y}, \rho_{(i)}), \\ \rho_{(i+1)} \leftarrow \rho_{(i)} + \gamma\beta(\mathbf{z} - F\mathbf{x}_{(i+1)} - G\mathbf{y}_{(i+1)}). \end{cases}$$

Here, $\gamma > 0$ is a constant. The ADM procedure has been proved to converge to the global optimal solution under quite broad conditions [11]. We now apply ADM to the various optimization problems encountered in Algorithms NVLCLUST and DICTCLUST. The ADM procedure stops when the relative change in the objective function is less than a given constant. In practice few iterations are needed to reach good results. Due to lack of space all proofs are omitted and can be found in the full version of the paper.

ADM for Detection Stage. Let $\mathbb{R}_+$ be the set of positive real numbers. In the detection stage of Algorithm NVLCLUST for each document $\mathbf{p}_j$, we solve the following program

$$\min_{\mathbf{x}\in\mathbb{R}^k} \|\mathbf{p}_j - A_{t-1}\mathbf{x}\|_1 + \lambda\|\mathbf{x}\|_1 \text{ s.t. } \mathbf{x} \ge 0. \quad (10)$$

This can be written equivalently as

$$\min_{\mathbf{x}\in\mathbb{R}_+^k, \mathbf{e}\in\mathbb{R}^m} \|\mathbf{e}\|_1 + \lambda\|\mathbf{x}\|_1 \text{ s.t. } \mathbf{e} = \mathbf{p}_j - A_{t-1}\mathbf{x}. \quad (11)$$

Then the augmented Lagrangian form of (11) is

$$\mathcal{L}(\mathbf{x}, \mathbf{e}, \rho) = \min_{\mathbf{x}\in\mathbb{R}_+^k, \mathbf{e}\in\mathbb{R}^m} \|\mathbf{e}\|_1 + \lambda\|\mathbf{x}\|_1 + \rho^\top(\mathbf{p}_j - A_{t-1}\mathbf{x} - \mathbf{e}) + (\beta/2)\|\mathbf{p}_j - A_{t-1}\mathbf{x} - \mathbf{e}\|_2^2 . \quad (12)$$

We now apply ADM to the above Lagrangian. Let us assume that we have $(\mathbf{x}_{(i)}, \mathbf{e}_{(i)}, \rho_{(i)})$, we construct $(\mathbf{x}_{(i+1)}, \mathbf{e}_{(i+1)}, \rho_{(i+1)})$ as follows. First, for a fixed $\mathbf{x}_{(i)}$ and $\rho_{(i)}$, we update $\mathbf{e}$ by solving

$$\min_{\mathbf{e}} \|\mathbf{e}\|_1 + \rho_{(i)}^\top(\mathbf{p}_j - A_{t-1}\mathbf{x}_{(i)} - \mathbf{e}) + \frac{\beta}{2}\|\mathbf{p}_j - A_{t-1}\mathbf{x}_{(i)} - \mathbf{e}\|_2^2.$$

LEMMA 6.1. *The minimum value of the above optimization is attained by setting* $\mathbf{e} = \textit{soft}(\mathbf{p}_j - A_{t-1}\mathbf{x}_{(i)} + \rho_{(i)}/\beta, 1/\beta)$.

The above lemma gives the derivation for $\mathbf{e}_{(i+1)}$. Now, for a fixed $\mathbf{e}_{(i+1)}$ and $\rho_{(i)}$ a simple manipulation shows that we can obtain $\mathbf{x}$ that minimizes (12) by solving the following

$$\min_{\mathbf{x}\in\mathbb{R}_+^k} \lambda\|\mathbf{x}\|_1 + (\beta/2)\|\mathbf{p}_j - A_{t-1}\mathbf{x} - \mathbf{e}_{(i+1)} + \rho_{(i)}/\beta\|_2^2. \quad (13)$$

However, instead of solving (13) exactly, we approximate it by

$$\min_{\mathbf{x}\in\mathbb{R}_+^k} \lambda\|\mathbf{x}\|_1 + \beta\left(\mathbf{g}_{(i)}^\top(\mathbf{x} - \mathbf{x}_{(i)}) + (1/2\tau)\|\mathbf{x} - \mathbf{x}_{(i)}\|_2^2\right), \quad (14)$$

[1]The subscript (i) denotes the ith iteration of the ADM procedure.

where $\tau > 0$ is a proximal parameter and

$$\mathbf{g}_{(i)} = \frac{\partial}{\partial \mathbf{x}} \|\mathbf{p}_j - A_{t-1}\mathbf{x} - \mathbf{e}_{(i+1)} + \rho_{(i)}/\beta\|_2^2 \text{ at } \mathbf{x} = \mathbf{x}_{(i)}$$
$$= A_{t-1}^\top (A_{t-1}\mathbf{x}_{(i)} + \mathbf{e}_{(i+1)} - \mathbf{p}_j - \rho_{(i)}/\beta).$$

The above approach belongs to the class of proximal gradient methods in optimization [33, 38]. We use it in the context of a composite function with a smooth and a non-smooth part, where the gradient is computed only based on the smooth part [33].

LEMMA 6.2. *The minimum value of* (14) *is attained by setting* $\mathbf{x} = \max\{\mathbf{x}_{(i)} - \tau \mathbf{g}_{(i)} - (\lambda\tau)/\beta, 0\}$.

The above lemma gives the derivation for $\mathbf{x}_{(i+1)}$. Now given fixed $\mathbf{x}_{(i+1)}$ and $\mathbf{e}_{(i+1)}$, we update multiplier ρ as $\rho_{(i+1)} = \rho_{(i)} + \gamma\beta(\mathbf{p}_j - A_{t-1}\mathbf{x}_{(i+1)} - \mathbf{e}_{(i+1)})$. The update equations are summarized in Figure 2. The following theorem shows the convergence of the ADM.

THEOREM 6.3. *Let* $\tau, \gamma > 0$ *satisfy* $\tau\lambda_{\max}(A) + \gamma < 2$, *where* $\lambda_{\max}(A)$ *is the maximum singular value of* A. *For any fixed* $\beta > 0$, *the sequence* $\{(\mathbf{e}_{(i)}, \mathbf{x}_{(i)}, \rho_{(i)})\}$ *generated by Figure 2 from any starting* $\{(\mathbf{e}_{(1)}, \mathbf{x}_{(1)}, \rho_{(1)})\}$ *converges to* $\{(\tilde{\mathbf{e}}, \tilde{\mathbf{x}}, \tilde{\rho})\}$ *where* $\tilde{\mathbf{e}}, \tilde{\mathbf{x}}$ *is the optimum solution to* (11).

ADM for Dictionary Learning Stage. In the dictionary learning stage, we solve (8) which can be rewritten as

$$\min_{A,\Gamma} \|\Gamma\|_1 + \lambda\|X_{\leq t}\|_1 \text{ s.t. } \Gamma = P_{\leq t} - AX_{\leq t}, A \geq 0.$$

Since $\|X_{\leq t}\|_1$ is a constant term in the objective, we can ignore that term and look at the following optimization problem

$$\min_{A,\Gamma} \|\Gamma\|_1 \text{ s.t. } \Gamma = P_{\leq t} - AX_{\leq t}, A \geq 0. \quad (15)$$

The augmented Lagrangian form of (15) is

$$\mathcal{L}(A, \Gamma, \Delta) = \min_{A \in \mathbb{R}_+^{m\times k}, \Gamma \in \mathbb{R}^{m\times n}} \|\Gamma\|_1 \quad (16)$$
$$+ \langle \Delta, P_{\leq t} - AX_{\leq t} - \Gamma\rangle + (\beta/2)\|P_{\leq t} - AX_{\leq t} - \Gamma\|_{\text{fro}}^2,$$

where $\Delta \in \mathbb{R}^{m\times n}$ is a multiplier and $\beta > 0$ is a penalty parameter. The ADM equations for this stage are a generalization (matrix version) of the ADM equations derived for the detection stage. Figure 2 summarizes these equations. We use A_{t-1} as warm restart for computing A_t. Similar to Theorem 6.3 one can also show the convergence of these ADM equations.

ADM for Algorithm DICTCLUST**.** We now derive the ADM equations for solving (7). We solve (7) by doing an alternative minimization over R and S.

$$R^{(j)} = \text{argmin}_{R\geq 0} \|\text{Nvl}_t - RS^{(j-1)}\|_1 + \lambda\|S^{(j-1)}\|_1, \quad (17)$$
$$S^{(j)} = \text{argmin}_{S\geq 0} \|\text{Nvl}_t - R^{(j)}S\|_1 + \lambda\|S\|_1. \quad (18)$$

We use ADM to solve both (17) and (18). The derivation of the ADM equations are quite similar to that done for detection and dictionary learning stages, so we summarize the results in Figure 3.

Remark about Parallelization. The optimization problems encountered in detection stage, dictionary learning stage, and Algorithm DICTCLUST can all be trivially parallelized. For detection stage, we could invoke ADM for each $\mathbf{p}_j \in P_t$ in parallel. For, the dictionary learning stage, we can rewrite the $\|P_{\leq t} - AX_{\leq t}\|_1$ as $\sum_{i=1}^m \|P_{\leq t}(i) - A(i)X_{\leq t}\|_1$ where $P_{\leq t}(i)$ and $A(i)$ are the ith rows in $P_{\leq t}$ and A, respectively. Since the terms in the summation are independent of each other, we can solve them in parallel using a modification of the ADM equations used for the detection stage. The right hand side of Figure 2 is a matrix version of this parallel optimization over rows of A. A similar idea also works for alternative minimization used in Algorithm DICTCLUST.

7. EMPIRICAL EVALUATION

In this section, we first empirically evaluate our approach on publicly-available labeled datasets from news streams and newsgroups. We then present a study of our approach applied to Twitter data from a PR campaign.

7.1 Evaluation Metrics

For the purpose of evaluation, we assume that documents in the corpus have been identified with a set of topics. For simplicity, we assume that each document is tagged with a single, most dominant topic that it associates with which we refer to as the *true topic* for that document.

We use variations of standard IR measures like pairwise precision, recall, and F1 score [24]. Given P_t, the set of documents arriving at time t, let $\text{TNvl}_t \subseteq P_t$ be the set of true novel documents in P_t (i.e., TNvl_t contains documents belonging to true emerging topic clusters). Let $\mathcal{C}_t$ be the set of system generated emerging topic clusters at time t, and let $\mathcal{T}_t$ be the true emerging topic clusters at time t. Note that clusters in $\mathcal{T}_t$ are formed over documents in TNvl_t, whereas the clusters in $\mathcal{C}_t$ are formed over documents in $\text{Nvl}_t \subseteq P_t$, and TNvl_t may not be equal to Nvl_t.

We define our evaluation metrics over the novel documents. Pairwise precision (Prec_{nvl}) is the number of pairs of documents that are in the same cluster in both $\mathcal{T}_t$ and $\mathcal{C}_t$ divided by the number of pairs of documents that are in the same cluster in $\mathcal{C}_t$. Pairwise recall (Rec_{nvl}) is the number of pairs of documents that are in the same cluster in both $\mathcal{T}_t$ and $\mathcal{C}_t$ divided by the number of pairs of documents that are in the same cluster in $\mathcal{T}_t$. Pairwise F1 (F1_{nvl}) is the harmonic mean of Prec_{nvl} and Rec_{nvl}. The following example illustrates these definitions. Let $P_t = \{p_1, p_2, \ldots, p_9\}$, $\text{TNvl}_t = \{p_3, p_4, p_5\}$, and $\text{Nvl}_t = \{p_3, p_4, p_5, p_6, p_7\}$. Let $\mathcal{T}_t = \{(p_3, p_4, p_5)\}$ (i.e., $\mathcal{T}_t$ contains one cluster made up of p_3, p_4, p_5), and $\mathcal{C}_t = \{(p_3, p_4, p_6), (p_5, p_7)\}$. Then,

$$\text{Prec}_{\text{nvl}} = \frac{|\{(p_3, p_4)\}|}{|\{(p_3, p_4), (p_3, p_6), (p_4, p_6), (p_5, p_7)\}|} = \frac{1}{4}$$
$$\text{Rec}_{\text{nvl}} = \frac{|\{(p_3, p_4)\}|}{|\{(p_3, p_4), (p_3, p_5), (p_4, p_5)\}|} = \frac{1}{3}$$

7.2 Baseline Approaches

We compare the performance of our algorithm against three alternative approaches we created, which are based on combining nearest neighbor and K-Means algorithms with dictionary learning. We describe these baselines below:

NN-KM: To detect novel documents, we use the nearest neighbor approach used by the UMass FSD system [2], which is one of the best performing system for this task [29]. As in the UMass system, we use cosine distance as a similarity measure and a TF-IDF weighted document representation. Every document in P_t whose cosine distance to its nearest neighbor in $P_{\leq t-1}$ is below some η is marked as novel. We build on this algorithm to get a baseline for emerging topic detection, by running a K-Means clustering with cosine distance (a.k.a. Spherical K-Means) on the documents marked novel. We use Spherical K-Means, as it is a well-established approach to clustering high-dimensional text data [12].

DICT-KM: The second baseline is a modification of our dictionary based scheme. We use a dictionary learning approach to detect

ADM EQUATIONS FOR (17)

Input: $S^{(j-1)} \in \mathbb{R}^{k_1 \times |\mathrm{Nvl}_t|}, \mathrm{Nvl}_t \in \mathbb{R}^{m \times |\mathrm{Nvl}_t|}, \beta, \tau, \gamma > 0$

1. $R^{(j)}_{(1)} \leftarrow R^{(j-1)}, \Gamma_{(1)} \leftarrow \mathrm{Nvl}_t - R^{(j-1)} S^{(j-1)}, \Delta_{(1)} \leftarrow 0$
2. **for** $i = 1, 2, \ldots,$*convergence* **do**
   a) $\Gamma_{(i+1)} \leftarrow \mathrm{soft}(\mathrm{Nvl}_t - R^{(j)}_{(i)} S^{(j-1)} + \Delta_{(i)}/\beta, 1/\beta)$
   b) $G_{(i)} \leftarrow (R^{(j)}_{(i)} S^{(j-1)} + \Gamma_{(i+1)} - \mathrm{Nvl}_t - \Delta_{(i)}/\beta) S^{\top}_{(j-1)}$
   c) $R^{(j)}_{(i+1)} \leftarrow \max\{R^{(j)}_{(i)} - \tau G_{(i)}, 0\}$
   d) $\Delta_{(i+1)} \leftarrow \Delta_{(i)} + \gamma\beta(\mathrm{Nvl}_t - R^{(j)}_{(i+1)} S^{(j-1)} - \Gamma_{(i+1)})$

ADM EQUATIONS FOR (18)

Input: $R^{(j)} \in \mathbb{R}^{m \times k_1}, \mathrm{Nvl}_t \in \mathbb{R}^{m \times |\mathrm{Nvl}_t|}, \lambda, \beta, \tau, \gamma > 0$

1. $S^{(j)}_{(1)} \leftarrow S^{(j-1)}, \Gamma_{(1)} \leftarrow \mathrm{Nvl}_t - R^{(j)} S^{(j-1)}, \Delta_{(1)} \leftarrow 0$
2. **for** $i = 1, 2, \ldots,$*convergence* **do**
   a) $\Gamma_{(i+1)} \leftarrow \mathrm{soft}(\mathrm{Nvl}_t - R^{(j)} S^{(j)}_{(i)} + \Delta_{(i)}/\beta, 1/\beta)$
   b) $G_{(i)} \leftarrow {R^{(j)}}^{\top} (R^{(j)} S^{(j)}_{(i)} + \Gamma_{(i+1)} - \mathrm{Nvl}_t - \Delta_{(i)}/\beta)$
   c) $S^{(j)}_{(i+1)} \leftarrow \max\{S^{(j)}_{(i)} - \tau G_{(i)} - (\lambda\tau)/\beta, 0\}$
   d) $\Delta_{(i+1)} \leftarrow \Delta_{(i)} + \gamma\beta(\mathrm{Nvl}_t - R^{(j)} S^{(j)}_{(i+1)} - \Gamma_{(i+1)})$

Figure 3: ADM equations for dictionary learning used in Algorithm DICTCLUST**.**

Phase	*No. of old clusters*	*No. of docs from old clusters*	*No. of new clusters*	*No. of docs from new clusters*	$\mathrm{F1}_{\mathrm{nvl}}$ (our)	$\mathrm{F1}_{\mathrm{nvl}}$ (NN-KM)	$\mathrm{F1}_{\mathrm{nvl}}$ (DICT-KM)	$\mathrm{F1}_{\mathrm{nvl}}$ (NN-DICT)
TDT2 Corpus								
1	18	1390	2	63	0.868	0.694	0.781	0.859
2	20	2256	6	139	0.349	0.324	0.364	0.338
3	26	669	3	45	0.784	0.650	0.730	0.721
4	29	2526	1	140	0.163	0.130	0.146	0.155
Avg.					**0.541**	**0.450**	**0.505**	**0.518**
20 NewsGroup Corpus								
1	6	572	2	783	0.652	0.555	0.630	0.626
2	8	769	2	991	0.586	0.536	0.561	0.519
3	10	964	2	1182	0.655	0.579	0.673	0.654
4	12	1160	2	1366	0.667	0.566	0.620	0.588
5	14	1359	2	1565	0.696	0.576	0.679	0.583
6	16	1552	2	1412	0.731	0.565	0.599	0.681
7	18	1713	2	1714	0.705	0.561	0.598	0.613
Avg.					**0.670**	**0.563**	**0.623**	**0.610**

Table 1: Results for TDT2 and 20 Newsgroups corpora. We set $k = 100$. The $\mathrm{F1}_{\mathrm{nvl}}$ value reported is the maximum $\mathrm{F1}_{\mathrm{nvl}}$ obtained for each phase by varying ζ, η in the interval $[0, 1]$. In Phase 0 of TDT2 corpus 2166 documents from 18 clusters and in Phase 0 of 20 Newsgroups corpus 1718 documents from 6 clusters are used to initialize the dictionary, respectively.

novel documents (this can be done by invoking Algorithm NVLCLUST without Step 3) and then run a Spherical K-Means clustering on these novel documents to create emerging topic clusters.

NN-DICT: The third baseline is also a modification of our dictionary based scheme. We first use the nearest neighbor approach (explained above) to detect novel documents and then run Algorithm DICTCLUST on these novel documents to create emerging topic clusters.

We implemented all the algorithms in Matlab. In all our experiments $k = 100$ and $\lambda = 1/100$. We noticed that varying these parameters does affect the evaluation metrics (e.g., increasing the dictionary sizes leads to better scores, but comes at a cost of increased running times). However, due to space constraints, we postpone a discussion of these results to the full version of this paper. The parameters of ADM are fixed as $\beta = 10$, $\tau = 1/50$, and $\gamma = 1.618$ (these are chosen in consultation with [38] and Theorem 6.3 for faster convergence).

7.3 TDT2 and 20 Newsgroups Datasets

We use two standard labeled datasets to evaluate the performance of our proposed algorithm. We start by describing these datasets and our experimental setup.

Our first dataset is the NIST topic detection and tracking (TDT2) corpus[2] which consists of news stories in the first half of 1998. For our evaluation, we use a set of 9,394 documents represented over 19,528 terms and spread over 27 weeks. These documents are partitioned into 30 human-labeled topics. We introduce the documents from the 27 weeks in 5 different phases. In the zeroth phase, we introduce all the documents between weeks 1 to 5 and these documents are used for initializing the dictionary A_0. We do so by running Algorithm NVLCLUST (without the Step 3) in an offline fashion (as in we alternate between the detection stage and dictionary learning stage for each document). In the first phase, we introduce all the documents between weeks 6 to 7 and run Algorithm NVLCLUST on these documents with dictionary A_0. In the second phase, we introduce all the documents between weeks 8 to 13 and run Algorithm NVLCLUST on these documents with dictionary A_1 (outputted by the first phase). We repeat the same steps for the third phase (between weeks 14 to 17) and fourth phase (between weeks 18 to 27). The reason for choosing such asymmetric time intervals is to make sure that at least one new topic cluster gets introduced in each phase.

As our second dataset we use the 20 Newsgroups corpus[3]. The corpus contains 18,774 articles distributed among 20 clusters where each cluster is a Usenet group. For our experiments, we use a vocabulary of 10,000 terms selected based on frequency. We do a set of controlled experiments on this corpus. Again, we introduce the documents in phases. Documents within each cluster are tempo-

[2] http://www.itl.nist.gov/iad/mig/tests/tdt/1998/

[3] http://people.csail.mit.edu/jrennie/20Newsgroups/

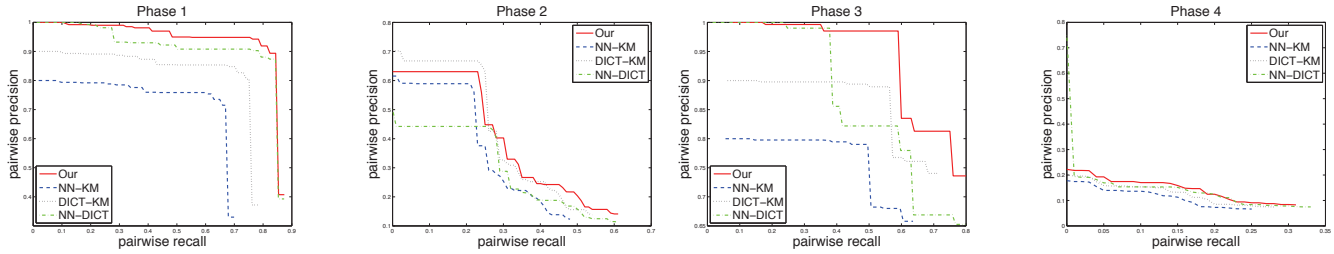

Figure 4: Interpolated pairwise precision-recall curves for TDT2, obtained by varying ζ and η between $[0.05, 0.15]$.

rally ordered, and we use this temporal ordering to introduce the documents. At the end of Phase $i-1$, we have documents from some (old) clusters, and in Phase i we introduce a mixture of documents, some coming from these old clusters and some belonging to new clusters; and see how well our algorithm performs in detecting these new clusters. We begin Phase 0 with documents sampled from 6 randomly chosen clusters. In each subsequent phase, we introduce documents from 2 new clusters. The numbers of documents from old and new clusters added at each phase are presented in Table 1.

For our baselines with K-Means clustering, we run the algorithm 8 times (with random initialization for centroids) and take the best result. For each phase, we set both k_1 (number of topics to be generated by Algorithm DICTCLUST) and K (number of clusters to be generated by K-Means) to be equal to 10. We vary the threshold $0 \leq \zeta \leq 1$ to find the threshold where $\mathrm{F1}_{\mathrm{nvl}}$ is maximized for our algorithm (similarly, for the DICT-KM algorithm with threshold ζ, and for the NN-KM and NN-DICT algorithms with threshold η).

Table 1 presents the maximum $\mathrm{F1}_{\mathrm{nvl}}$ for both datasets (obtained by varying ζ, η). Our algorithm always outperforms all the three baselines. For TDT2, our algorithm gives on average 16.9% improvement in $\mathrm{F1}_{\mathrm{nvl}}$ score over the NN-KM, 6.7% improvement over DICT-KM, and 4.3% improvement over NN-DICT. For 20 Newsgroups, we notice on average 16.0% improvement over NN-KM, 7.0% improvement over DICT-KM, and 9.0% improvement over NN-DICT. While the results shows that we get a big improvement over NN-KM, the improvements over DICT-KM and NN-DICT are only moderate. Since, both DICT-KM and NN-DICT use in parts our dictionary learning algorithm, the results suggest that using dictionary learning could lead to an improvement in the $\mathrm{F1}_{\mathrm{nvl}}$ score over just using nearest neighbor/clustering techniques. Furthermore, our algorithm consistently outperforms DICT-KM, and NN-DICT consistently outperforms NN-KM, indicating that a dictionary-based clustering scheme results in better clustering than Spherical K-Means.

Note that, the $\mathrm{F1}_{\mathrm{nvl}}$ scores are lower than what one would encounter in "typical clustering" applications. Intuitively, this happens because we evaluate these metrics only over the novel documents, which are far fewer in number than the documents from the old clusters; and each missed document (from $\mathrm{TNvl}_t \setminus \mathrm{Nvl}_t$) affects $\mathrm{Rec}_{\mathrm{nvl}}$ combinatorially because of the pairwise mistakes that they lead to (similarly, each wrongly added document from $\mathrm{Nvl}_t \setminus \mathrm{TNvl}_t$ severely affects $\mathrm{Prec}_{\mathrm{nvl}}$).

In Figure 4, we present interpolated (pairwise) precision-recall curves [24] for the TDT2 dataset, obtained by varying the threshold ζ, η. We see that the precision-recall curve of our algorithm mostly dominates the curves of the baseline algorithms, which illustrates the superiority of our algorithm over different operating points. The curves for the 20 Newsgroups corpus are similar and are omitted for lack of space.

7.4 Novel Document Detection Evaluation

The performance of our approach relies on first, accurately identifying novel documents, and then appropriately clustering these document. Results in the previous section show that we do well in the overall task of emerging topic detection, but in order to get a better understanding of the strengths of our approach, we investigate performance on the sub-task of novel document detection. Given a set of documents, this task is to classify each document as either novel (positive) or non-novel (negative). True labeling is created by labeling the documents from any of the emerging topic clusters as positive, and the documents from any of the existing true topics as negative. For evaluating this classification task, we use Area Under Curve (AUC) of the ROC curve [24]. As a baseline, we use the nearest neighbor (NN) approach explained in Section 7.2. As mentioned earlier, this NN approach is one of the best performing systems for FSD, a task that is very similar to novel document detection.

The results, presented in Table 2, show that for this sub-task our dictionary-based approach and the nearest neighbor approach have almost the same performance. Even though the AUCs are near identical, the sets of documents that the two approaches mark as novel are quite different. This is evidenced by the superior performance of our approach over NN-DICT, which uses the same approach for the second sub-task of clustering novel documents into emerging topics (see Table 1). In summary, while our dictionary-based approach is picking novel documents with the same accuracy as NN, it is more likely to pick documents from the same emerging topic.

	TDT2		**20 Newsgroups**	
Phase	*Our*	*NN*	*Our*	*NN*
1	0.982	0.982	0.941	0.942
2	0.926	0.933	0.941	0.942
3	0.976	0.961	0.880	0.880
4	0.844	0.850	0.898	0.902
5			0.915	0.917
6			0.917	0.921
7			0.919	0.920
Avg.	**0.932**	**0.932**	**0.916**	**0.918**

Table 2: Performance, in AUC, on novel document detection.

7.5 Watson Jeopardy Twitter Data

In Section 7.3, we evaluated our approach on labeled benchmark data sets. Here, we analyze how well our approach performs in practice, when applied to Twitter data. In particular, we look at the Twitter discussion around IBM's Watson DeepQA system.[4] In February 2011, the Watson DeepQA system participated on the

[4]http://www.ibm.com/watson

Jeopardy! game-show, competing against humans; and the event was televised over 3 days: Feb 14, 15, and 16. We used Twitter's search API to collect all tweets relevant to the participation of Watson in Jeopardy! from Feb 1 – 16; where tweets were judged relevant by keyword and pattern filters. We used a sample set of 8,434 tweets from Feb 1 – 14, to initialize our dictionary. Then, we tested our system on detecting emerging topics on a sample of 5,199 tweets from Feb 16. The entire corpus of tweets had a vocabulary of size 1,139, after removing common stop words.

We give as input to Algorithm NVLCLUST the learned dictionary and the tweets of Feb 16. Instead of presenting the entire range of parameter values, here we pick $k_1 = 15$ and $\zeta = 0.5$ to illustrate the clusters identified at a particular setting. From the topic clusters outputted by our algorithm, in Table 3, we present three selected ones. These clusters were manually inspected to identify the dominant topic in them. Note that, these three topics are emerging, as the date of their occurrence is Feb 15 or later, whereas we learn our initial dictionary only on tweets till Feb 14.

The fourth and fifth columns in Table 3 show the size of the novel clusters detected, and the number of tweets that were judged, by a human annotator, to be relevant to the identified dominant topic in these clusters. The sixth column shows the top keywords for each topic. These are selected by looking at the atoms corresponding to these topics in the dictionary matrix produced by Algorithm DICTCLUST and by picking the top 3 words on which these atoms have the largest mass. The first selected cluster contains tweets on Watson's Final Jeopardy mistake on Feb 15th, which subsequently received much media coverage. The second and third selected clusters contain tweets related to two influential news articles on Feb 16. Note that, large fractions of tweets in these clusters are relevant to the dominant topic, indicating high coherence of the identified clusters. Table 4 presents three sample tweets from each of the three topics; which illustrates that our approach can cluster together tweets that are clearly on the same topic, even if syntactically quite different. Being able to automatically comb through tens of thousands, and potentially millions, of documents, to identify such emerging topics of discussion amongst consumers is of significant value to PR and marketing efforts.

8. RELATED WORK

Several existing approaches to identify *hot topics* or *trends* in social media are based on the frequency of mentions of terms, such as Twitscoop.com, Trendistic.com, Twopular.com, and trends on Twitter.com. While high frequency of terms may be a good indicator of popularity, it does not necessarily identify new or emerging trends. Instead, comparing the *relative* frequency of occurrence of terms and phrases in the current time period to the occurrences in the past, is likely to identify more topical phrases. Tomokiyo and Hurst [32] propose such an approach for extracting key-phrases based on statistical language models, which they apply to 20 Newsgroups data. Similarly, Cataldi *et al*. [25] present an approach in which they identify emergent keywords, which have been extensively used in a given time period, but not in previous ones. They find words that frequently co-occur with each emergent word, and report these together as emerging topics. Glass *et al*. [14] go a step further, from emerging words to tracking emerging *memes* – "distinctive phrases which propagate relatively unchanged." Their work focuses on building models to predict which memes will spread widely. Our work differs from the above approaches by going beyond unigrams and n-grams, to identifying novel clusters of similar documents, which provides a richer characterization of topics. Furthermore, we distinguish emerging, novel topics from merely popular or well-known ones.

9. CONCLUSION

User-generated content is increasingly playing a pivotal role as the source for breaking news and developments, as well as shaping opinions on a variety of matters ranging from products to policies. In this paper, we presented a dictionary learning based framework for detecting emerging topics in social media and related streams. The dictionary learning formulation naturally combines ideas from robustness, sparsity, and non-negative matrix factorization for analysis of streaming text. The overall framework was divided into two stages—first, determining novel documents in the stream, and subsequently identifying cluster structure among the novel documents. The objective functions in each stage were optimized using the alternating directions method, which can be easily parallelized for further scalability. Empirical evaluation on a variety of datasets illustrate the effectiveness of the proposed framework.

One can envision several directions for future work based on the proposed framework. While the current work uses fixed sized dictionaries, using adaptive dictionaries whose size changes based on the set of active or emerging topics may be more desirable in certain applications. While learning the dictionary A_t, the current work uses the entire historical data $P_{\leq t}$. A fully online version which only maintains sufficient statistics of historical data may be more scalable for real world streams. Further, from an optimization perspective, one may be able to use accelerated gradient descent and related proximal methods [33] to further speed up the alternating directions method. Finally, while the focus of the current work was on detecting emerging topics in text streams, similar ideas can be developed for other domains (such as healthcare, climate sciences) where detection of novel signal streams is of interest.

Acknowledgments

We would like to thank Richard Lawrence for many fruitful discussions. We would also like to thank Estepan Meliksetian and Phaneendra Divakaruni for collecting the Jeopardy Twitter dataset. This research is continuing through participation in the Anomaly Detection at Multiple Scales (ADAMS) program sponsored by the U.S. Defense Advanced Research Projects Agency (DARPA) under Agreement Number W911NF-11-C-0200. AB was supported in part by NSF CAREER award IIS-0953274, and NSF grants IIS-1029711, IIS-0916750, and IIS-0812183.

10. REFERENCES

[1] M. Aharon, M. Elad, and A. Bruckstein. The K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE TSP*, 54(11):4311–4322, 2006.

[2] J. Allan. *Topic Detection and Tracking: Event-based Information Organization*. Springer, 2002.

[3] E. Amaldi and V. Kann. On the Approximability of Minimizing Nonzero Variables or Unsatisfied Relations in Linear Systems. *Elsevier TCS*, 209(1-2):237–260, 1998.

[4] R. Basri and D. Jacobs. Lambertian Reflectance and Linear Subspaces. *IEEE TPAMI*, pages 218–233, 2003.

[5] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet Allocation. *JMLR*, 3:993–1022, 2003.

[6] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning*, 2011.

[7] A. Bruckstein, D. Donoho, and M. Elad. From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images. *SIAM review*, 51(1):34–81, 2009.

Id	*Topic/Event*	*Date of Occurrence*	*Size of Detected Cluster*	*Tweets Relevant to Topic in the Cluster*	*Top Topic Keywords*
1	Watson's Final Jeopardy Mistake [34]	Feb 15, 2011	118	118	trebek, toronto, check
2	News Article [30]	Feb 16, 2011	110	82	people, amazed, google
3	News Article [18]	Feb 16, 2011	100	70	speed, creators, advantage

Table 3: Emerging Topics on Feb 16, 2011 in Watson Jeopardy Twitter Archive.

Id	*Sample Tweets from each of the* 3 *Emerging Topics shown in Table 3*
1	Watson on Jeopardy! Day Two: The Confusion Over an Airport Clue US Cities: Its largest airport was named for a WWII hero; its 2nd for a WWII battle. IBM Watson: What is Toronto Why Watson answered Toronto to "Its largest airport was named for a World War II hero; its second largest, for...
2	Search Engine Land: Could Google Play Jeopardy Like IBM's Watson? SEO::: Could Google Play Jeopardy Like IBM's Watson?: Like many people, I was amazed to watch IBM's Watson super... Could Google Play Jeopardy Like IBM's Watson?: Like many people, I was amazed to watch IBM's Watson supercompute...
3	Arstechnica: Creators: Watson has no speed advantage as it crushes humans in Jeopardy New on ArsTechnica Creators: Watson has no speed advantage as it crushes humans in Jeopardy ... Technology Creators: Watson has no speed advantage as it crushes humans in Jeopardy

Table 4: Some tweets from the 3 **emerging topics detected on Feb 16, 2011. We omit the tiny urls appearing in the tweets.**

[8] E. Candes, X. Li, Y. Ma, and J. Wright. Robust Principal Component Analysis. abs/0912.3599, 2009.

[9] E. Candes and P. Randall. Highly Robust Error Correction by Convex Programming. *IEEE TIT*, 54(7):2829–2840, 2008.

[10] S. Chen, D. Donoho, and M. Saunders. Atomic Decomposition by Basis Pursuit. *SIAM SISC*, 20(1), 1999.

[11] P. Combettes and J. Pesquet. Proximal Splitting Methods in Signal Processing. Arxiv preprint arXiv:0912.3522, 2009.

[12] I. S. Dhillon and D. S. Modha. Concept Decompositions for Large Sparse Text Data using Clustering. *Machine Learning*, 42(1):143–175, 2001.

[13] D. Donoho. For most Large Underdetermined Systems of Equations, the Minimal L1-norm Near-solution Approximates the Sparsest Near-solution. *Communications on Pure and Applied Mathematics*, 59(7):907–934, 2006.

[14] K. Glass and R. Colbaugh. Toward Emerging Topic Detection for Business Intelligence: Predictive Analysis of Meme Dynamics. *CoRR*, 2010.

[15] T. Hofmann. Probabilistic Latent Semantic Analysis. In *UAI*, pages 289–296, 1999.

[16] B. J. Jansen, M. Zhang, K. Sobel, and A. Chowdury. Twitter Power: Tweets as Electronic Word of Mouth. *J. Am. Soc. Inf. Sci. Technol.*, 60, 2009.

[17] R. Jennaton, J. Mairal, G. Obozinsky, , and F. Bach. Proximal Methods for Sparse Hierarchical Dictionary Learning. *ICML*, pages 487–494, 2010.

[18] C. Johnston. Creators: Watson has no speed advantage as it crushes humans in Jeopardy. http://bit.ly/fmfmwz.

[19] Q. Ke and T. Kanade. Robust L1 Norm Factorization in the Presence of Outliers and Missing Data by Alternative Convex Programming. In *CVPR*, pages 739–746, 2005.

[20] D. Lee and H. Seung. Learning the Parts of Objects by Non-negative Matrix Factorization. *Nature*, 1999.

[21] H. Lee, A. Battle, R. Raina, and A. Ng. Efficient Sparse Coding Algorithms. *NIPS*, 19:801–808, 2007.

[22] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online Learning for Matrix Factorization and Sparse Coding. *JMLR*, 2010.

[23] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative Learned Dictionaries for Local Image Analysis. In *CVPR*, pages 1–8, 2008.

[24] C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[25] C. Mario, D. C. Luigi, and S. Claudio. Emerging Topic Detection on Twitter based on Temporal and Social Terms Evaluation. In *IWMDM*, pages 1–10, 2010.

[26] P. Melville, V. Sindhwani, and R. Lawrence. Social Media Analytics: Channeling the Power of the Blogosphere for Marketing Insight. In *Proc. of the WIN*, 2009.

[27] K. Min, Z. Zhang, J. Wright, and Y. Ma. Decomposing Background Topics from Keywords by Principal Component Pursuit. In *CIKM*, pages 269–278, 2010.

[28] B. Olshausen and D. Field. Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

[29] S. Petrović, M. Osborne, and V. Lavrenko. Streaming First Story Detection with Application to Twitter. In *ACL*, 2010.

[30] D. Sullivan. Could Google Play Jeopardy Like IBMs Watson? http://selnd.com/hL5kvY.

[31] R. Tibshirani. Regression Shrinkage and Selection via the Lasso. *JRSS*, 58(1):267–288, 1996.

[32] T. Tomokiyo and M. Hurst. A Language Model approach to Keyphrase Extraction. In *ACL Workshop on Multi-word Expressions*, pages 33–40, 2003.

[33] P. Tseng. Aprroximation Accuracy, Gradient Methods, and Error Bound for Structured Convex Optimization. *Mathematical Programming, Series B*, 125:263–295, 2010.

[34] Wikipedia. Watson (computer). http://bit.ly/mTwstz.

[35] J. Wright and Y. Ma. Dense Error Correction Via L1-Minimization. *IEEE TIT*, 56(7):3540–3560, 2010.

[36] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma. Robust Face Recognition via Sparse Representation. *IEEE TPAMI*, 2008.

[37] A. Y. Yang, S. S. Sastry, A. Ganesh, and Y. Ma. Fast L1-minimization Algorithms and an Application in Robust Face Recognition: A Review. In *ICIP*, 2010.

[38] J. Yang and Y. Zhang. Alternating Direction Algorithms for L1-Problems in Compressive Sensing. Arxiv, 2009.

[39] Y. Zhang, A. d'Aspremont, and L. E. Ghaoui. *SparsePCA: Convex Relaxations, Algorithms and Applications*. Springer, 2010.