Dynamic Auto-Encoders for Semantic Indexing

Piotr Mirowski
Courant Institute of Mathematical Sciences
New York University
textttmirowski@cs.nyu.edu

Marc'Aurelio Ranzato
Department of Computer Science
University of Toronto
ranzato@cs.toronto.edu

Yann LeCun
Courant Institute of Mathematical Sciences
New York University
yann@cs.nyu.edu

Abstract

We present a new algorithm for topic modeling, text classification and retrieval, tailored to sequences of time-stamped documents. Based on the auto-encoder architecture, our nonlinear multi-layer model is trained stage-wise to produce increasingly more compact representations of bags-of-words at the document or paragraph level, thus performing a semantic analysis. It also incorporates simple temporal dynamics on the latent representations, to take advantage of the inherent structure of sequences of documents, and can simultaneously perform a supervised classification or regression on document labels. Learning this model is done by maximizing the joint likelihood of the model, and we use an approximate gradient-based MAP inference. We demonstrate that by minimizing a weighted cross-entropy loss between histograms of word occurrences and their reconstruction, we directly minimize the topic-model perplexity, and show that our topic model obtains lower perplexity than Latent Dirichlet Allocation on the NIPS and State of the Union datasets. We illustrate how the dynamical constraints help the learning while enabling to visualize the topic trajectory. Finally, we demonstrate state-of-the-art information retrieval and classification results on the Reuters collection, as well as an application to volatility forecasting from financial news.

1 Introduction

We propose in this article a new model for sequences of observations of discrete data, specifically word counts in consecutive (or time-stamped) text documents, such as online news, recurrent scientific publications or periodic political discourses. We build upon the classical bag-of-words approach, ignoring the syntactic dependencies between words; we focus instead on the text semantics by looking at vocabulary distributions at the paragraph or document level. Our method can automatically discover and exploit sequences of low-dimensional latent representations of such documents. Unlike most latent variable or topic models, our latent representations can be simultaneously constrained both with simple temporal dependencies and with document labels. One of our motivations is the sentiment analysis of streams of documents, and has interesting business applications, such as ratings prediction. In this work, we predict the volatility of a company's stock, by capturing the opinion of investors manifested in online news about that company.

Simple word counts-based techniques, such as the Term Frequency - Inverse Document Frequency (TF-IDF) remain a standard method for information retrieval (IR) tasks (for instance returning documents of the relevant *category* in response to a query). TF-IDF can also be coupled with a classifier (such as an SVM with linear or Gaussian kernels) to produce state-of-the-art text classifiers [1], [2]. We thus show in Results section 3.3 how our low-dimensional document representation measures up to TF-IDF or TF-IDF + SVM benchmarks on information retrieval and text categorization tasks.

Plain TF-IDF relies on a high-dimensional representation of text (over all V words in the vocabulary) and compact representations are preferable for index lookup because of storage and speed issues. A candidate for such low-dimensional representations is Latent Semantic Analysis (LSA) [3], which is based on singular value decomposition (SVD). Alternatively, one can follow the dimensionality reduction by independent components analysis (ICA), to obtain statistically independent latent variables [4]. Unfortunately, because they perform lossy compression and are not trained discriminatively w.r.t. the task, SVD and ICA achieve worse IR performance than the full TF-IDF.

Instead of linear dimensionality reduction, our approach is to build *auto-encoders*. An auto-encoder is an architecture trained to provide with a latent representation (*encoding*) of its input, thanks to a nonlinear *encoder* module and an associated *decoder* module. Auto-encoders can be stacked and made into a deep (multi-layer) neural network architecture [5][6][7][8]. A (semi-)supervised deep auto-encoder for text has been introduced in [9] and achieved state-of-the-art classification and IR.

There are three crucial differences between our model and Ranzato and Szummer's [9]. First of all, our model makes use of latent variables. These variables are inferred though the minimization of an energy (over a whole sequence of documents) that involves the reconstruction, the temporal dynamics, the code prediction, and the category (during supervised learning), whereas in [9], the codes are simply computed deterministically by feed-forward encoders (their inference does not involve energy minimization and relaxation). It is the same difference as between a dynamic Bayesian net and a simple feed-forward neural net. Secondly, our cross-entropy loss function is specifically constructed to minimize topic model perplexity, unlike in [9]. Instead of merely predicting word counts (through an un-normalized Poisson regression), we predict the smoothed word distribution. This allows us to actually model topics probabilistically. Lastly, our model has a hierarchical temporal structure, and because of its more flexible nature, is applicable to a wider variety of tasks.

Several auto-encoders have been designed as probabilistic graphical models in order to model word counts, using binary stochastic hidden units and a Poisson decoder [10], [8] or a Softmax decoder [11]. Despite not being a true graphical model when it comes to the inference of the latent representation, our own auto-encoder approach is also based on the Softmax decoder, and, as explained in Methods section 2.3, we also do take into account varying document lengths when training our model. Moreover, and unlike [10], [8] or [11], our method is supervised and discriminative, and further allows for a latent dynamical model.

Another kind of graphical models specifically designed for word counts is topic models. Our benchmark is the Latent Dirichlet Allocation [12], which defines a posterior distribution of M topics over each document, and samples words from sampled topics using a word-topic matrix and the latent topic distribution. We also considered its discriminative counterpart, Supervised Topic Models [13] with a simple linear regression module, on our financial prediction task (in Results section 3.4). We show in Results section 3.1 that we managed to achieve lower perplexity than LDA.

Some topic models have introduced dynamics on the topics, modeled as Gaussian random walks [14], or Dirichlet processes [15]. A variant to explicit dynamics consists in modeling the influence of a ”time” variable [16]. Some of those techniques can be expensive: in Dynamic Topic Models [14], there is one topic-word matrix per time step, used to model drift in topic definition. Moreover, inference in such topic models is intractable and replaced either by complex Variational Bayes, or by Gibbs sampling. Finally, all the above temporal topic models are purely generative.

The major problem with the Gaussian random walks underlying [14] is that they describe a smooth dynamic on the latent topics. This might be appropriate for domains such as scientific papers, where innovation spreads gradually over time [14], but might be inexact for political or financial news, with sudden ”revolutions”. For this reason, we considered Laplace random walks, that allow for ”jumps”, and illustrated in section 3.2 the trajectory of the U.S. State of the Union speeches.

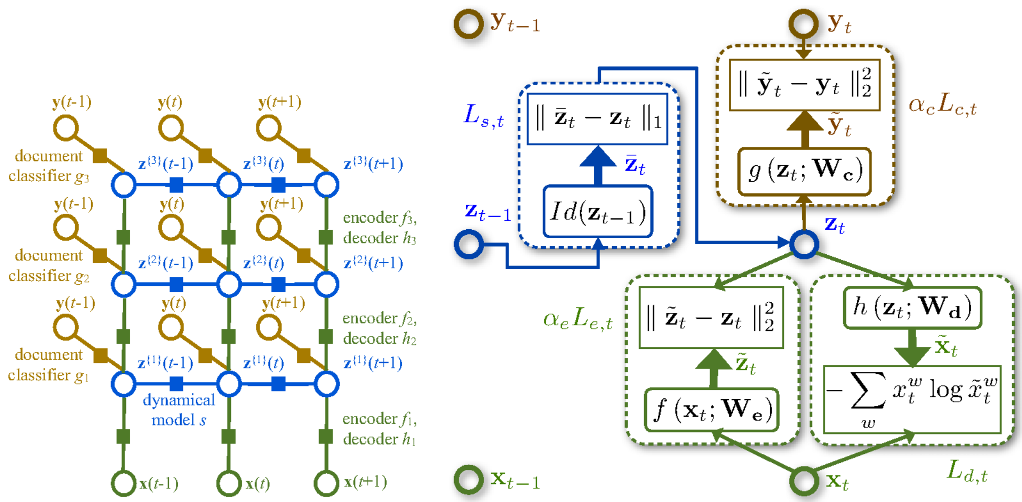
2 Methods

For each text corpus, we assume a vocabulary of V unique tokens, which can be words, word stems, or named entities[1]. The input to the system is a V-dimensional bag-of-words representation $\mathbf{x}_i$ of each document i, in the form of a histogram of word counts $n_{i,v}$, with $N_i = \sum_{v=1}^{V} n_{i,v}$. To avoid zero-valued priors on word occurrences, probabilities $\mathbf{x}_i$ can be smoothed with a small coefficient β (here set to 10^{-3}): $\mathbf{x}_i \equiv \frac{n_{i,v}+\beta}{N_i+\beta V}$.

[1]We built a named-entity recognition pipeline, using libraries from the General Architecture for Text Engineering (http://gate.ac.uk), and relying on gazetteer lists enriched with custom lists of company names.

2.1 Auto-Encoder architecture on bag-of-words histograms

The goal of our system is to extract a hierarchical, compact representation from very high-dimensional input vectors $\mathbf{X} = \{\mathbf{x}_i\}_i$ and potential scalar or multivariate labels $\mathbf{Y} = \{\mathbf{y}_i\}_i$. This latent representation consists in K layers $\mathbf{Z}^{\{k\}} = \{\mathbf{z}_i^{\{k\}}\}_i^{\{k\}}$ of decreasing dimensionality $V > M_1 > M_2 > \ldots > M_K$ (see Fig. 1a). We produce this representation using deep (multi-layer) auto-encoders [5][6][7][8] with additional dynamical constraints on the latent variables. Each layer of the auto-encoder is composed of modules, which consist in a parametric deterministic function plus an error (loss) term, and can be interpreted as conditional probabilities.



(a) 3-layer deep auto-encoder

(b) Layer 1

Figure 1: Left: Factor graph representation of the deep auto-encoder architecture with dynamical dependencies between latent variables. Right: Energy-based view of the first layer of the dynamic auto-encoder. The reconstruction factor comprises a decoder module h with cross-entropy loss $L_{d,t}$ w.r.t. word distribution $\{x_t^w\}_{w=1}^V$, and an encoder module f with Gaussian loss $L_{e,t}$, for a total factor's loss $\alpha_e L_{e,t} + L_{d,t}$. The latent variables $\mathbf{z}_t$ are averaged by time unit into $\mathbf{Z}_{t'-1}, \mathbf{Z}_{t'}, \ldots$, and the latter follow Gaussian or Laplace random walk dynamics defined by the dynamical factor and associated loss $\alpha_s L_{s,t'}$ (for simplicity, we assumed here 1 document for time unit t' and one for previous time unit $t'-1$). There is an optional supervised classification/regression module g (here with a Gaussian regression loss $\alpha_c L_{c,t}$).

The *encoder* module of the k-th layer transforms the inputs (word distribution $\mathbf{x}_i$ if $k = 1$) or variables from the previous layer $\mathbf{z}_i^{\{k-1\}}$ into a latent representation $\mathbf{z}_i^{\{k\}}$. The encoding function $f_k(\mathbf{z}_i^{\{k-1\}}) + \epsilon_i = \mathbf{z}_i^{\{k\}}$ or $f_k(\mathbf{x}_i) + \epsilon_i = \mathbf{z}_i^{\{1\}}$is parametric (with parameters noted $\mathbf{W_e}$). Typically, we use the classical $tanh$ sigmoid non-linearity, or a sparsifying non-linearity $x^3/(x^2+\theta)$ where θ is positive[2]. The mean square loss term ϵ_i represents Gaussian regression of latent variables.

Conversely, there is a linear *decoder* module (parameterized by $\mathbf{W_d}$ on the same k-th layer that reconstructs the layer's inputs from the latent representation $h_k(\mathbf{z}_i^{\{k\}}) + \delta_i = \mathbf{z}_i^{\{k-1\}}$, with a Gaussian loss δ_i. Layer 1 is special, with a normal encoder but with a Softmax decoder h_1 and a cross-entropy loss term, as in [11]:

$$\bar{x}_i^v = \frac{\exp(\mathbf{W}_{\mathbf{d}v}\mathbf{z}_i^{\{1\}})}{\sum_{v'}\exp(\mathbf{W}_{\mathbf{d}v'}\mathbf{z}_i^{\{1\}})} \quad (1)$$

The dynamical module of the k-th layer corresponds to a simple random walk from a document at time step t to next document at time step $t+1$: $\mathbf{z}_{t+1}^{\{k\}} = \mathbf{z}_t^{\{k\}} + \eta_i$. The error term η_i can be either a

[2]The sparsifying nonlinearity is asymptotically linear but shrinks small values to zero. θ should be optimized during the learning, but we decided, after exploratory analysis on training data, to set it to a fixed value of 10^{-4}

sum of squared element-wise differences (L_2-norm) between the consecutive time-unit averages of latent codes of documents (i.e. a Gaussian random walk, that enforces smooth dynamics), or a sum of absolute values of those element-wise differences (L_1-norm, i.e. Laplace random walk).

There can be multiple documents with the same timestamp, in which case, there should be no direct constraints between $\mathbf{z}_{a,t}$ and $\mathbf{z}_{b,t}$ of two documents a and b sharing the same time-stamp t. In the case of such hierarchical temporal dynamics, we define a dynamic between consecutive values of the averages $< \mathbf{z} >_t$ of the latent variables from same time-unit documents (for a set I_t of N_t articles published on the same day t, each average is defined as $< \mathbf{z} >_t \equiv 1/N_t \sum_{i \in I_t} \mathbf{z}_i$). The intuition behind the time-specific averages of topics is that they capture the topic "trend" for each time stamp (e.g. year for NIPS proceedings or for State-of-the-Union speeches).

Finally, there is a classification/regression module g_k that classifies k-th layer latent variables. Typically, we considered multi-variate logistic regression (for classification problems) or linear regression with logistic loss or Gaussian loss, respectively.

Those models can be learned in a greedy, sequentially layer-wise approach [5], by considering each layer as an approximated graphical model (see Fig. 1b) and by minimizing its negative log-likelihood using an Expectation Maximization (EM) procedure with an approximate maximum-a-posteriori inference (see next sub-section 2.2). We finally prove how our learning procedure minimizes the topic model perplexity (sub-section 2.3).

2.2 Dynamic factor graphs and Maximum A Posteriori approximation

Factor graphs [18] are a convenient formalism for graphical models. They express the joint likelihood of all visible and hidden variables: each factor, that is connected to a subset of variables, models the conditional dependencies between those variables. The likelihood of the model is the product of likelihoods of each factor, and in log space, this product becomes a sum.

Using maximum a posteriori (MAP) inference, we do not marginalize over the latent variables by integrating them out. Instead, we minimize the negative log-likelihood of the model by finding the configuration of latent variables that minimizes a loss function that corresponds to an unnormalized negative log-likelihood. We decide to ignore the partition function, which is intractable, and in order not to reach degenerate solutions, we impose constraints on the latent variables during inference. This enables us to consider potentially nonlinear factors (modules). Our gradient-based EM algorithm is a coordinate descent on the log-likelihood over the sequence:

$$L(\mathbf{X}, \mathbf{Y}; \mathbf{W}) = \min_{\mathbf{Z}} \{L_d(\mathbf{X}, \mathbf{Z}; \mathbf{W}_d) + \alpha_c L_c(\mathbf{Z}, \mathbf{Y}; \mathbf{W}_c) + \alpha_e L_e(\mathbf{X}, \mathbf{Z}; \mathbf{W}_e) + \alpha_s L_s(\mathbf{Z})\} \quad (2)$$

Each iterative inference (E-step) and learning (M-step) consists in a full relaxation w.r.t. latent variables or parameters, like in the original EM algorithm. We use simple gradient descent to minimize negative log-likelihood loss w.r.t. latent variables, and conjugate gradient with line search to mimize L w.r.t. parameters. Because each relaxation is until convergence and done separately, everything else being fixed, the various hyperparameters for learning the modules can be tuned independently, and the only subtlety is in the choice of the weights α_c, α_e and α_s. The α_* coefficients control the relative importance of the encoder, decoder, dynamics and supervised modules in the total energy, and they can be chosen by cross-validation.

We add an additional Laplace prior on the weights and latent variables (using L_1-norm regularization, and multiplying learning rates by $\lambda_w = \lambda_z = 10^{-4}$). Finally, we normalize the decoder to unit column weights as in the sparse decomposition [19]. Because we initialize the latent variable by first propagating the inputs of the layer through the encoder, then doing a relaxation, the relaxation always gives the same latent variables for given parameters, inputs and labels.

As a variation on a theme, we can directly encode $\mathbf{x}_i$ using the encoders $f_1, f_2, \ldots, f_K$, like in [9], in order to perform fast inference (e.g. for information retrieval or for prediction, as we did on experiments in sections 3.3. or 3.4).

2.3 Minimizing topic model perplexity

In the field of topic models, the perplexity measures the difficulty of predicting documents after training model $\mathbf{\Omega}$, and is evaluated on held out test sets. Under an independence assumption, and on

Algorithm 1 EM-type learning of the latent representation at layer k of the Dynamic Factor Graph

if $k = 1$ **then**
  Use bag-of-words histograms $\mathbf{X}$ as inputs to the first layer
else
  Use M_{k-1}-dimensional hidden representation $\mathbf{Z}^{\{k-1\}}$ as inputs to layer k
end if
Initialize the latent variables $\mathbf{Z}^{\{k\}}$ using M_k-dimensional ICA
while $epoch \leq n_{epochs}$ **do**
  // M-step on the full training sequence:
  Optimize the softmax $(k = 1)$ or Gaussian decoder h_k by minimizing loss L w.r.t. $\mathbf{W}_d$
  Optimize the nonlinear encoder f_k by minimizing loss L w.r.t. $\mathbf{W}_e$
  Optimize the logistic classifier or linear regressor g_k by minimizing loss L w.r.t. $\mathbf{W}_c$
  // E-step on the full training sequence:
  Infer the latent variables $\mathbf{Z}^{\{k\}}$ using the **encoder** f_k, and record associated loss $L'(epoch)$
  Continue inference of $\mathbf{Z}^{\{k\}}$ by minimizing loss L (Equation 7) w.r.t. $\mathbf{Z}^{\{k\}}$ (relaxation)
  if encoder-only loss $L'(epoch)$ is the lowest so far **then**
    Store the "optimal" parameters $\{\mathbf{W}_e, \mathbf{W}_d, \mathbf{W}_c\}$
  end if
end while
Infer $\mathbf{Z}^{\{k\}}$ using "optimal" parameters and the encoder f_k only
Optional: continue the inference by minimizing loss L w.r.t. $\mathbf{Z}^{\{k\}}$

a set $\{\mathbf{w}_i\}_{i=1}^T$ of T documents, containing N_i words each, perplexity is defined in [12] as:

$$P \equiv p\left(\{\mathbf{w}_i\}_{i=1}^T|\Omega\right)^{-\frac{1}{\sum_{i=1}^T N_i}} = \exp\left(-\frac{\sum_{i=1}^T \log p\left(\mathbf{w}_i|\mathbf{\Omega}\right)}{\sum_{i=1}^T N_i}\right) \tag{3}$$

In most topic models, each document i is associated with a latent representation θ_i (e.g. the multinomial posterior distribution over topics in LDA), and one assumes the document to be a bag of N_i conditionally independent words $\mathbf{w}_i = \{w_{i,n}\}_{n=1}^{N_i}$. Hence, the marginal distribution of $\mathbf{w}_i$ is:

$$p\left(\mathbf{w}_i|\mathbf{\Omega}\right) = \int_{\theta_i} p\left(\theta_i|\mathbf{\Omega}\right)\left(\prod_{n=1}^{N_i} p(w_{i,n}|\theta_i, \mathbf{\Omega})\right) d\theta_i \approx \prod_{n=1}^{N_i} p(w_{i,n}|\tilde{\theta}_i, \mathbf{\Omega}) \tag{4}$$

In LDA, the topic assignment distribution $\tilde{\theta}_i$ is inferred for each document d from observed word occurrences, using variational inference [12] or Gibbs sampling [20]. In our maximum-a-posteriori approach, we replace the full distribution over θ_i by a delta distribution with a mode at $\tilde{\theta}_i$ that maximizes the likelihood. We rewrite equation (4):

$$\log p\left(\mathbf{w}_i|\tilde{\theta}_i, \mathbf{\Omega}\right) = \sum_{n=1}^{N_i} \log p(w_{i,n}|\tilde{\theta}_i, \mathbf{\Omega}) = N_i \sum_{v=1}^{V} \frac{n_{i,v}}{N_i} \log p(v|\tilde{\theta}_i, \mathbf{\Omega}) \tag{5}$$

By defining the empirical conditional distribution of words in document d as $p_i(v) \equiv \frac{n_{i,v}}{N_i}$, which we substitute in (5), and by noting the model conditional distribution as $q_i(v) \equiv p(v|\tilde{\theta}_i.\mathbf{\Omega})$, equation (5) become proportional to the cross-entropy between the empirical and the model conditional distributions over words for document i: $H\left(p_i(v), q_i(v)\right) = -\sum_v p_i(v) \log q_i(v)$. Given this derivation and MAP approximation, the perplexity of our topic model can be expressed in terms of a weighted sum of cross-entropies (the weights are proportional to the documents' lengths):

$$P \approx \tilde{P} = \exp\left(\frac{1}{\sum_{i=1}^T N_i} \sum_{i=1}^{T} N_i H(p_i, q_i)\right) \tag{6}$$

Minimizing LDA perplexity (3) is equivalent to minimizing the negative log-likelihood of the model probabilities of words in all documents, i.e. to a maximum likelihood solution. This is what we do in our approximate maximum-a-posteriori (MAP) solution, by minimizing a weighted cross-entropy loss (7) with respect to both the model parameters $\mathbf{\Omega}$ and the latent representations $\{\theta_i\}_{i=1}^T$. Using an unnormalized latent document representation $\mathbf{z}_i$ (instead of LDA's simplex θ_i), and in lieu of model distribution q_i, our model reconstructs a V-dimensional *output* vector $\bar{\mathbf{x}}_i$ of positive values

Table 1: Test set perplexity on NIPS articles using 100-30-10-2 DAE with 3 different dynamical models: each layer of DAE is compared to LDA with the same number M of latent topics.

M	**LDA**	**DAE**$_{\alpha_s=0}$, no dynamics	**DAE**$_{\alpha_s=1}$, L_1 dynamics	**DAE**$_{\alpha_s=1}$, L_2 dynamics
100	657	**518**	**522**	**522**
30	760	**698**	**695**	**695**
10	848	903	909	960

summing to 1 through the sequence of decoding functions (we write it $\tilde{\mathbf{x}}_i = h(\mathbf{z}_i)$). However, instead of integrating over the latent variables as in (4), we minimize the *reconstruction loss* (7) over the hidden representation. For a document i, the cross-entropy $-\sum_v x_{i,v} \log \tilde{x}_{i,v}$ is measured between the actually observed distribution $\mathbf{x}_i$, and the predicted distribution $\tilde{\mathbf{x}}_i$.

$$L_d \left(\{p_i\}_{i=1}^T ; \mathbf{\Omega} \right) \quad \equiv \quad \min_{\{q_i\}_i} \left(\sum_{i=1}^T N_i H(p_i, q_i) \right) = \min_{\tilde{\mathbf{x}}_i} \left(- \sum_{i=1}^T \mathbf{x}_i^T \log \tilde{\mathbf{x}}_i \right) \tag{7}$$

3 Results

3.1 Perplexity of unsupervised Dynamic Auto-Encoders

In order to evaluate the quality of Dynamic Auto-Encoders as topic models, we performed a comparison of DAE vs. Latent Dirichlet Allocation. More specifically, for a 100-30-10-2 DAE architecture, we compared the perplexity of 100-topic LDA vs. the perplexity of the 1st layer of the DAE, then the perplexities of 30-topic LDA vs. the 2nd DAE layer, and so on for 10-topic and 2-topic LDA.

The dataset, consisting in 2483 NIPS articles published from 1987 to 2003, was separated into a training set (2286 articles until 2002) and a test set (197 articles from 2003). We kept the top $V = 2000$ words with the highest TF-IDF score. 100-, 30-, 10- and 2-topic LDA "encodings" [12] were performed using Gibbs sampling inference[3] [20]. Our 100-30-10 DAE with encoding weight $\alpha_e = 0$ achieved lower perplexity than LDA on the first two layers (see Table 1). We also empirically compared L_1 or L_2 dynamical penalties vs. no dynamics ($\alpha_s = 0$). There was little difference between the three types on the first 2 layers. However, L_1 norm (Laplace) dynamics instead of (Gaussian) L_2 helped for further layers, as on the 3rd layer, no dynamics and L_1 decreased perplexity by 10%. Moreover, L_1 allowed a large "jump" in topic space between 2001 and 2002.

3.2 Visualization of topic trajectories

We reproduced a study on the U.S. State of the Union speeches from [15]. We selected the top $V = 2000$ common words and named entities (using the same method as in section 3.4), and defined a training set consisting in 17,350 paragraphs from 196 yearly speeches through 1989, and a test set of 1965 paragraphs from 19 speeches (1990 to 2010). After training a 100-30-10-2 DAE with L_1 dynamics, we visualized the 2D topic trajectories taken by the yearly averages of latent variables on the 4th layer, and compared them with a non-dynamical DAE (same architecture) and a 3-topic LDA (with 2 degrees of freedom). As Figure 2 shows, using dynamics on the latent variables during the E-step inference helps to produce a latent representation (in the space spanned by the two hidden units/topics) that can be useful when we expect a dynamical structure in the data.

The latter 2D latent representation provided us with a historical interpretation. It appeared that the five strongest discontinuities in the L_1 norm between 4-th hidden topics were, in that order: Harry Truman (1946), Ronald Reagan (1982, inaugural), Andrew Jackson (1829, inaugural), Woodrow Wilson (1913, inaugural) and Franklin Roosevelt (1934, inaugural), and on the test data, George W Bush (2001, inaugural), William Clinton (1996) and Barack Obama (2009, inaugural), which seemed to confirm historical intuitions.

[3] Using Xuan-Hieu Phan's GibbsLDA++ package, available at http://gibbslda.sourceforge.net/, we trained Gibbs-sampled sLDA for 2000 iterations, with standard priors $\alpha = 50/M$ and $\beta = 20/V$

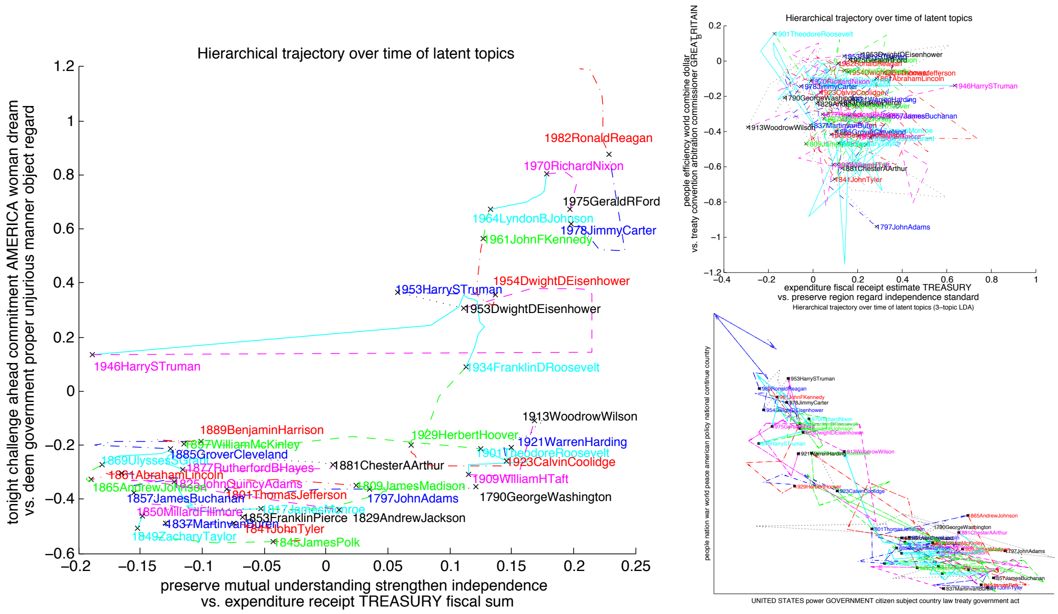


Figure 2: Left: 2D "trajectory" in the latent space of the 4th layer of the 100-30-10-2 DAE, with dynamical weight $\alpha_s = 1$. On each axis, "vs." opposes the words at the two extremes of that axis. Latent variables were inferred per paragraph and averaged by year. Top right: same figure for a DAE without dynamics ($\alpha_s = 0$). Bottom right: same figure for a 3-topic LDA (2 degrees of freedom).

3.3 Text categorization and information retrieval

The standard Reuters-21578 "ModApt" collection[4] contains 12,902 financial articles published by the Reuters news aggregator, split into 9603 train and 3998 test samples. Each article belongs to zero, one or more categories (in this case, the type of commodity described), and we considered the traditional set of the 10 most populated categories (note that both [10] and [9] mistakenly interpreted that collection as a dataset of 11,000 train and 4000 test single-class articles). We generated stemmed word-count matrices from raw text files using the Rainbow toolbox[5], selecting the top $V = 2000$ word stems with respect to an information gain criterion, and arranging articles by publication date.

To our knowledge, the state-of-the-art classification technique on that set remains Support Vector Machines with linear or Gaussian kernels [1]. We used the standard SVM software packages *liblinear*[6] and *libsvm*[7], and performed a five-fold cross-validation to select the hyperparameters (regularization C, Gaussian width γ) through exhaustive search on a coarse, then on a fine grid. We compared a 100-30-10-2 DAE with a neural network encoder[8], to TF-IDF, ICA [4], SVD [3], LDA [12], [20], and auto-encoders [9]. The area under the precision recall curve for retrieval of same-category documents (interpolated as in [21]) by TF-IDF was 0.51, and 0.543 using 10-dimensional ICA on TF-IDF (which was by far the best among unsupervised techniques). After optimizing the inference weights on the training data ($\alpha_e = 1, \alpha_s = 1$ and $\alpha_c = 100$), we achieved 0.89, 0.63, 0.78 and 0.75 on the 1st, 2nd, 3rd and 4th layers, which compares with 0.87, 0.93, 0.89 and 0.70 for auto-encoders in [9] with the same architecture. In [9] though there is no relaxation step on the latent variables during learning, only direct inference, which might help to better train the encoder for IR.

For the multi-class classification task, we computed multi-class precision, recall and F_1 using micro and macro-averaging [1], [2]. Using an SVM with linear kernel trained on the latent variables, we

[4] Available at Alessandro Moschitti's webpage: http://dit.unitn.it/∼moschitt/corpora.htm

[5] Andrew McCallum's toolbox is available at http://www.cs.cmu.edu/∼mccallum/bow/rainbow/

[6] Available at http://www.csie.ntu.edu.tw/∼cjlin/liblinear/

[7] Available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm/

[8] At each layer, we decided for the NN to have twice as many hidden nodes as outputs. Effectively, our architecture could be called 200-100-60-30-20-10-4-2, but the inference/relaxation, as well as supervised label information and dynamical information, were used only on the 100, 30, 10 and 2-dimensional hidden layers.

Table 2: Prediction of median-adjusted log-volatility from 2008 financial news about the Dow 30 components, using a linear regression fit on bag-of-words-derived representations ($V = 2000$). We report the coefficient of determination R^2 on the test set (articles published after June 30).

TF-IDF+lin	**TF-IDF+SVR**	**ICA$_{100}$**	**LDA$_{100}$**	**DAE$_{100}$**	**DAE$_{30}$**	**DAE$_{10}$**	**DAE$_{2}$**
0.267	**0.287**	0.219	0.134	0.261	0.263	0.271	**0.283**

achieved a macro-averaged F_1 score of 0.85 on the first 3 layers, and a micro-average F_1 of 0.92 on the same layers, which compares to 0.85 and 0.92 respectively using [9] and to 0.83 and 0.91 for each score using TF-IDF. We can therefore claim that we are close to the state of the art for information retrieval and text classification.

3.4 Prediction of stock market volatility using financial news from Bloomberg

There is some evidence in recent history that financial markets (over)react to public information. In a simpler setting, one can restrict this observation to company-specific news and associated stock movements, quantified with volatility σ^2. The problem of volatility forecasting from financial news has been formulated as a supervised text categorization problem and addressed in an intra-day setting [22], where it was proved that the arrival of some "shock" news about an asset j impacted its volatility (by switching to a "high-volatility" mode) for a duration of at least 15 min. In this article, we tried to solve a slightly more difficult problem than in [22], by considering the volatility $\sigma^2_{j,t}$ estimated from daily stock prices[9] of a company j. We normalized volatility dividing it by the median volatility across all companies j on that same day, then taking its logarithm: $y_{j,t} = \log \sigma^2_{j,t} - \log \bar{\sigma}^2_t$. Using the Bloomberg Professional service, we collected over 90,000 articles, published between January 1 and December 31, 2008, on 30 companies that were components of the Dow Jones index on June 30, 2008. We extracted each document's time stamp and matched it to the log-volatility measure $y_{j,t}$ at the earliest following market closing time. Common words and named entities were extracted, and numbers (dollar amounts, percentages) were binned. In order to make the problem challenging, we split the dataset into 51,362 test articles (after July 1, 2008, in a crisis market) and 38,968 training articles (up to June 30, 2008, corresponding to a more optimistic market).

As we report in Table 2, our DAE with a 100-30-10-2 architecture, $tanh$ encoders f and linear regressors g achieves, on all layers, a higher coefficient of determination R^2 on the test set than linear encodings (M-dimensional ICA on TF-IDF) or probabilistic topic models (M-topic LDA). Most importantly, the low-dimensional DAE latent representations (3rd and 4th layer) outperforms the full high-dimensional sparse representation (TF-IDF) with linear regression, and also matches the R^2 score of TF-IDF followed by nonlinear regression such as Support Vector Regression, which is very slow and expensive to train on this large Bloomberg dataset (90k training examples) sLDA [14] performed surprisingly poorly, with $R^2 < 0.1$, for 100, 30, 10 or 2 topics.

Finally, those compact representations of the early 2008 financial articles highlighted informative text features about stock market uncertainty: for instance, the two 2nd-layer hidden topics that were the most positive regressors of log-volatility had the following topic definitions (top 10 words): topic 1: *GM, sale, US, vehicle, BOEING, world, +20%, automaker, +10%, plant* and topic 2: *rating, credit, financial, information, debt, MOODY'S, FITCH, flow, security, AIG* .

4 Conclusion

We have introduced a new method for information retrieval, text categorization and topic modeling, that can be trained in a both purely generative and discriminative ways. It can give word probabilities per document, like in a topic model, and incorporates temporal dynamics on the topics. Moreover, learning and inference in our model is simple, as it relies on an approximate MAP inference and a greedy approach for multi-layer auto-encoders. This results in a few hours of learning time on moderately large text corpora, using unoptimized Matlab code. Our initial results on standard text collections are very promising. As further avenues of work, we are planning on optimizing the gradient-based algorithms for training individual components of the DAE.

[9] Stock market data were acquired at http://finance.yahoo.com. Volatility was estimated from daily open, close, high and low prices using the Yang & Zhang formula, available at http://www.sitmo.com/eq/417.

References

[1] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML*, 1998.

[2] F. Debole and F. Sebastiani. An analysis of the relative hardness of the reuters-21578 subsets. *Journal of the American Society for Information Science and Technology*, 56:584–596, 2005.

[3] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[4] T. Kolenda and L. Kai Hansen. Independent components in text. In *Advances in Independent Component Analysis*, 2000.

[5] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep belief networks. In *NIPS*, 2006.

[6] G.E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006.

[7] M.'A. Ranzato, Y. Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. In *NIPS*, 2007.

[8] R. Salakhutdinov and G. Hinton. Semantic hashing. In *ACM SIGIR Workshop on Information Retrieval and Applications of Graphical Models*, 2007.

[9] M.'A. Ranzato and M. Szummer. Semi-supervised learning of compact document representations with deep networks. In *ICML*, 2008.

[10] P.V. Gehler, A.D. Holub, and M. Welling. The rate adapting poisson model for information retrieval and object recognition. In *ICML*, 2006.

[11] R. Salakhutdinov and G. Hinton. Replicated softmax. In *ICML*, 2009.

[12] D.M. Blei, A.Y Ng, and M.I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[13] D.M. Blei and J.D. McAulife. Supervised topic models. In *NIPS*, 2007.

[14] D.M. Blei and J.D. Lafferty. Dynamic topic models. In *ICML*, 2006.

[15] I. Pruteanu-Malinici, L. Ren, J. Pailsey, E. Wang, and L. Carin. Hierarchical bayesian modeling of topics in time-stamped documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:996–1011, 2010.

[16] X. Wang and A. McCallum. Topics over time: A non-markov continuous-time model of topical trends. In *KDD*, 2006.

[17] N.N. Taleb. *The Black Swan: The Impact of the Highly Improbable*. Random House, New York, 2007.

[18] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[19] B.A. Olshausen and D.J. Field. Sparse coding with an overcomplete basis set: a strategy employed by v1? *Vision Research*, 37:3311–3325, 1997.

[20] T. Griffiths and M. Steyvers. Finding scientific topics. *PNAS*, 10(1):5228–5235, 2004.

[21] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *ICML*, 2006.

[22] C.S. Robertson, S. Geva, and R.C. Wolff. News aware volatility forecasting: Is the content of the news important? In *Sixth Australasian Data Mining Conference*, 2007.