TopicAE: A Topic Modeling Autoencoder

Miroslav Smatana, Peter Butka

Faculty of Electrical Engineering and Informatics, Department of Cybernetics and Artificial Intelligence
TechnicalUniversity of Košice
Letná 9, 040 01 Košice, Slovakia
e-mails: {miroslav.smatana, peter.butka}@tuke.sk

Abstract: In this paper, we propose TopicAE, a simple autoencoder designed to perform topic modeling with input texts. Topic modeling has grown in popularity especially in recent years with a large number of digital documents and contributions from social media available. These texts usually contain useful information and methods in the area of topic modeling, show novel approaches to their automatic summarization, browsing and searching. The main idea of topic modeling is to uncover hidden semantic structures from the input collection of texts. There are several topic models to extract standard topics from with their evolution through time and hierarchical structure of the topics. In this paper, we propose techniques known from the area of neural networks. Our TopicAE model can be applied to solve all the tasks mentioned above. The performance of the proposed model was also tested and showed that TopicAE could solve the topic modeling problem and out performed standard methods (Latent Semantic Indexing and Latent Dirichlet Allocation) according to evaluation metrics.

Keywords: autoencoder; deep learning; neural networks; topic modeling

1 Introduction

In the last one and a half decade, the internet and especially social media have become one of the powerful communication tools providing new possibilities for data gathering and analysis. The Internet is a source of the enormous amount of data in various forms (audio, video, text, etc.). In this paper, we focus on textual data, which usually contain useful information such as opinions and attitudes related to different people, organizations, products, world events, etc. This information can be used in several ways, mostly by organizations to increase their profit, for example:

- Launching of new products – when a company introduces a new product to the market, topic modeling can be used to track topics in which the

product is discussed, see details of the opinions, problems with the product, or which products are most competitive according to users.

- Crisis analysis –in the time of a war conflict it is possible to monitor how users perceive the current situation; it is possible to track the evolution of reactions, make adequate actions.
- Targeted marketing – tracking what people usually discuss and predict which product they might buy.
- Analysis and protection of reputation – this represents an opportunity to monitor social media to catch different contributions with negative or positive opinions on a company or person.
- Media – in this case, topic modeling helps to analyze, summarize, and visualize the news, search for reactions of people on them, their evolvement and sharing.

Currently, digital textual data play a key role in many tasks. However, due to their significant amount, it is difficult to find helpful information, especially in social media sources. Such needs lead to new methods for the extraction and processing of textual data. For that reason, the so-called topic modeling became a popular and powerful tool for the automatic semantic analysis of large collections of texts. It shows new ways of browsing, searching, and summarizing such collections. The main idea of topic modeling is to uncover hidden semantic structures (topics) in texts, where a topic is represented as a probability distribution over the fixed vocabulary of words (terms).

An example of topic modeling output (for example of article from news in Associated Press) can be in this form:
"The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Julliard School. Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important...", where every color in the text represents a different topic (for example red color - arts, green color - budgets, etc.).

There are already several approaches to topic modeling. Most of them focus on the analysis in the context of static corpora. Most of the standard methods are based on the recognition of topics as collections of terms and computation of their probabilities. Latent Semantic Indexing (LSI) [1] and Latent Dirichlet Allocation (LDA) [2] are well-known and often applied standard methods, which create a probabilistic model of topics from the input corpus of documents. However, in many problems of practical significance, a simple structure in the form of an extracted set of topics for the whole corpora is not enough. There are at least two interesting sub-tasks of topic modeling, which can be extracted within topics to make them more informative. The first is that changes in time or evolution of the topics are interesting, especially in the context of data streams such as social media. In this

case, it is essential to capture changes in the topic structure. The second is that in many cases it can be seen that some topic is more structured or complicated, and it has a hierarchical structure of subtopics.

Several methods, which extract the evolution or the hierarchical structure of topics, were also already introduced, and we describe some of them in the next section within related work. Standard methods share their main feature – all of them are generative probabilistic models. In this paper, we would like to introduce our model, which provides different non-probabilistic and non-generative approach based on neural networks.

In particular, the paper describes the proposed neural network layer, TopicAE (Topic AutoEncoder) which can be applied to solve the problem of building all three types of topic modeling tasks (basic topic model, the evolution of topics in time, the hierarchical structure of sub-topics). We also provide experiments with the selected datasets, where TopicAE is compared to standard topic models using several metrics.

The remainder of the present paper is structured as follows. In the next section, we present related work for each topic modeling problem with a more detailed description of the selected methods. In Section 3, we provide some necessary details on neural networks needed for the introduction of the proposed TopicAE approach in the following section. In Section 5, we present experiments with the selected dataset as well as comparison of TopicAE, standard topic modeling methods and selected neural network models.

2 Related Work

In this section, we focus on the selected methods applied to achieve the goals of particular topic modeling tasks. First, we start our discussion with a simple latent topic model which formalizes the basic ideas in the topic modeling. Next, we describe more advanced models which can capture time evolution of topics and hierarchical topic structures. We also mention some of the neural network approaches related to topic modeling tasks.

2.1. Latent Dirichlet Allocation

Latent Semantic Indexing (LSI) [1] can be considered as one of the first methods for topic analysis. Even though if it is not always perceived as a topic modeling method, it creates a base for probabilistic latent semantic analysis [3]. The basic ideas behind this approach lead to the most known topic modeling method - Latent DirichletAllocation (LDA) – first described by Blei et al. in [2]. LDA became the de-facto standard of topic modeling and is often used as a baseline method in comparisons with new approaches.

Before we give a brief detailed description of LDA, we could mention that there are also many LDA extensions such as Petterson et al. [4] or data stream extension [5]. Another type of topic modeling method is the HierarchicalDirichlet Process (HDP) [6], which became widely used as the core of present topic modeling methods.

Now we provide a short description of the LDA method (based on [2]) for mining of the standard topic model. LDA is a generative probabilistic model of corpus data. The main idea of LDA is that input documents are represented as random mixtures over latent topics, and each of these topics is characterized by a distribution over words. The basic terms are defined as:

- A word is a basic unit of discrete data and belongs to finite vocabulary indexed by $\{1, \ldots, V\}$. The v-the word in the vocabulary is represented by a V-vector of weights w so that $w^v = 1$, $w^u = 0$ and $u \neq v$.
- A document is a sequence of N words denoted by weights $w = (w_1, w_2, \ldots, w_N)$, where w_n is a weight for n-th word in a sequence.
- A corpus is a collection of M documents denoted by $D=\{d_1, d_2, \ldots, d_M\}$.

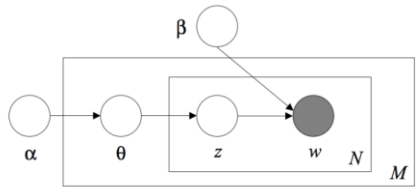

Figure 1

Probabilistic graphical representation of LDA model [2]. In this case, we have a collection of M documents represented by sequences of N words, which lead to topic models z. The whole process is controlled by parameters α and β.

LDA assumes the following generative process (graphical representation is shown in **Figure 1**) for each document in input corpus D:

1. Choose $N \sim Poisson$
2. Choose $\theta \sim Dir(\alpha)$
3. For every n-th word from N words:
    a. Choose a topic $z_n \sim Multinomial(\theta)$
    b. Choose a word w_n from $p(w_n|z_n,\beta)$, a multinomial probability conditioned on the topic z_n.

This model has several assumptions. First, dimensionality k of the Dirichlet distribution (and thus the dimensionality of z) over (k - 1) simplex is assumed known and fixed. Second, the word probabilities are parametrized by a $k \times V$ matrix β,where $\beta_{ij} = p(w^j = 1|z^i = 1)$. θ is k-dimensional Dirichlet random variable,

$\alpha$is a k-vector with components $\alpha_i > 0$. The α and β represent corpus level parameters and are sampled once in the process of processing a corpus. θ are document level variables and are sampled once per document. Variables z and w are word-levels and are sampled once for each word in each document.

2.2. Dynamic Topic Models

One can imagine that instead of simple topic modeling results, a more structured output from such approaches can be of interest for users. One of such extensions is related to time perspective in topic modeling and leads to the evolution of topics in time.

In this case, Blei and Lafferty [7] present a method called Dynamic Topic Models (DTM), which belongs to the family of probabilistic time series models and provides time evolution of topics in input collections of texts. This model works only with a discrete space model, so to resolve this problem of discretization Wang et al. present DTM extension called Continuous Time Dynamic Topic Models (cDTM) [8]. Moreover, Beykikhoshk et al. in [9] present a different approach to capturing topic time evolution based on the Hierarchical Dirichlet Process.

Differently to LDA, where documents are selected equivalently from the same set of topics, DTM approach supposes that input corpus is divided by time slice with K-component topic model and topics associated with slice t evolved from topics generated in slice t-1.

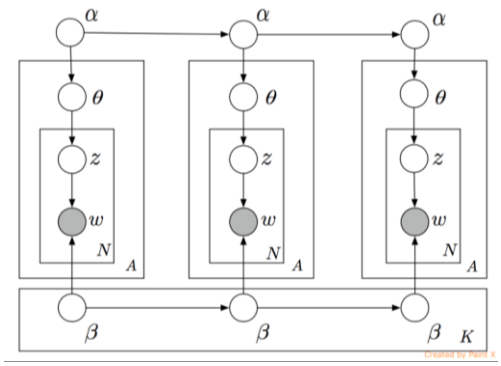

Figure 2

Graphical representation of probabilistic model known as DTM (Dynamic Topic Models) [7] used for modeling of topics evolution in time

Using the same notation as in LDA, DTM generative process for slice t is defined as follows (graphical model of DTM is given in [7]):

1. Draw topics $\beta_t \mid \beta_{t-1} \sim \mathcal{N}(\beta_{t-1}, \sigma^2 I)$
2. Draw $\alpha_t \mid \alpha_{t-1} \sim \mathcal{N}(\alpha_{t-1}, \delta^2 I)$

3. For each document:
    a. Draw $\eta \sim \mathcal{N}(\alpha_t, a^2 I)$
    b. For each word:
        i. Draw z ~ $Mult(\pi(\eta))$
        ii. Draw $w_{t,d,z} \sim Mult(\pi(\beta_{t,z}))$

Note that π maps the multinomial natural parameters to the mean parameters, and $\mathcal{N}$ represents an extension of the logistic normal distribution to time-series simplex data [7].

2.3. Hierarchical Topic Models

A different family of topic modeling approaches is related to methods which can capture hierarchical topic structure. In this case, a more detailed analysis of particular topics from higher levels leads to their subtopics with the result in the form of topic hierarchy. Different methods have already been developed to achieve such a goal, e.g., Blei et al. presented a method based on a nested Chinese restaurant process in [10], Hoffman described a cluster-based method [11], Smith et al. [12] provided the hierarchical version of LDA. From other approaches we can mention Pachinko Allocation Model (PAM) [13], Hierarchical Latent Tree Analysis (HLTA) [14] or a method presented in [15], where authors developed a hierarchical model based on HDP.

Now we describe one of these approaches, a generative probabilistic model for learning the hierarchical structure of topics as an extension of LDA based on nested processes [10]. This hierarchy is an *L*-level tree, where each node is associated with a topic. In this approach, Bayesian perspective is applied to the problem of topic hierarchy extraction. Here, hierarchies are random variables and these random variables are specified procedurally. It is based on the Chinese restaurant process (CRP) [10] and is defined as follows (using notation as for LDA):

1. Let c_1 be the root restaurant
2. For each level $l \in \{2,\ldots,L\}$:
    a. Draw a table from the restaurant c_{l-1}. Set c_l to be the restaurant referred to by table.
3. Draw an L-dimensional topic proportion vector θ from $Dir(\alpha)$
4. For each word $w \in \{1,\ldots,N\}$:
    a. Draw $z \in \{1,\ldots,L\}$ from $Mult(\theta)$
    b. Draw w_n from the topic associated with restaurant c_z.

Graphical representation of this hierarchical model is shown in Figure **3**. The node labeled *T* refers to a collection of an infinite number of *L*-level paths drawn from a nested CRP, γ, η are hyperparameters for T, β and distribution of *c* is defined by the nested Chinese restaurant process.

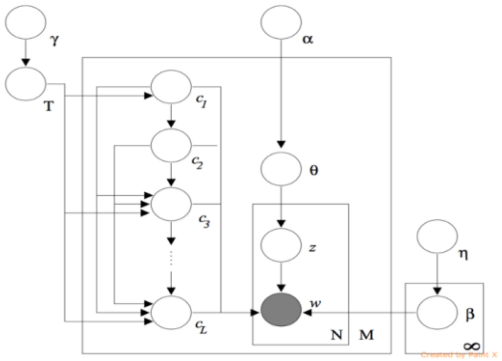

Figure 3
Graphical representation of hierarchical LDA topic model based on the nested Chinese restaurant process [10]

2.4. Neural Networks Approaches

Neural networks are techniques which are capable of automatically extracting a low dimensional representation of input data. Due to their growing popularity in recent years, these methods have become more popular in the field of natural language processing. Several works aimed to extract meaningful document representation (topics).

In their work, Salakhutdinov and Hinton [20] present a two-layer undirected graphical model called Replicated Softmax (RSM). Larochelle and Lauly [21] propose DocNADE, which is a neural autoregressive topic model that estimates the probability of observing a new word in the document by previously observed words in that document. As their results show, this approach outperforms the RSM model and solves RSM computational complexity on data with a large vocabulary. There are also other interesting models such as neural variational inference model NVDM [22], neural topic model NTM [23], k-competitive autoencoder KATE [24] or ProdLDA [26], which combine neural networks with the classic LDA model. In the field of topic evolution in time, Gupta et al. [25] present a model based on a recurrent neural network and replicated softmax to extract topical trends over time.

3 Preliminaries on Neural Networks

In this section, we shortly provide key properties of neural networks, which are necessary for the description of the proposed autoencoder TopicAE.Neural networks (NN) [16] is a computing system inspired by biological nervous systems. It is composed of a large number of highly interconnected elements called neurons, which cooperate to solve a specific problem. NN is also called a massive parallel processor model and like people can learn from examples and use gained knowledge in the future. To learn NN, we need training examples $(\boldsymbol{x}^{(i)},y^{(i)})$, where $i\in\{1,\ldots,n\}$ represents the index of i-th training example from our training set of n examples. Here, x represents a vector of input features and y represents output, which we want to predict (in this simplest case it is one value of prediction, but we can also have a vector of values in a more general case). For example, in the medical domain, every patient is a training example, where x is a vector of all measured values (symptoms), and y is the output value from {0,1} if the patient has or has not a disease.

3.1 Single Neuron and Simple Neural Network

To describe the basics of neural networks, we will start with the description of the simplest neural network, which consists of a single neuron. The structure of neuron is shown in Figure **4.**

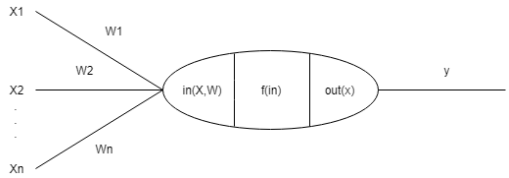

Figure 4

Structure of a single neuron in a neural network with inputs and weights represented by vectors x and w (producing aggregated input), activation function f and output y

A neuron is a computational unit, which has inputs - $\{x_1, x_2, ..., x_n\}$ and generates output y. A basic neuron consists of the followingparts:

- weights $\{w_1, w_2, ..., w_n\}$ - used to connect neurons within NN, bearers of information in NN,
- *in* - input to the neuron - is function of inputs $\{x_1, x_2, ..., x_n\}$. In most cases inputs are aggregated using sum function:

$$in = \sum_{j=1}^{n} w_j\, x_j \tag{1}$$

- *f(in)* - activation function - there are several types of activation functions, in this paper we use *f(in)* represented by sigmoid function:

$$f(in) = \frac{1}{1+exp(-in)} \tag{2}$$

- *out(x)* - output function, which is usually identical function - *out(x) = x*

A neural network is a structure, which connects many neurons, so that the output of one neuron is the input to another neuron. An example of a simple feed-forward NN is shown in Figure 5. L1 represents input layer, L2 hidden layer (output values of neurons in this layer are not available in a training set), a L3 output layer and $a^{(i)}_j$ represent the output of a *j*-th neuron in the *i*-th layer.

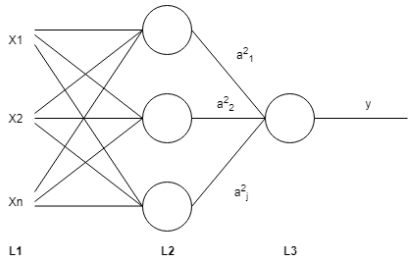

Figure 5

Structure of a single neuron in a neural network with inputs and weights represented by vectors x and w (producing aggregated input), activation function f and output y.

The number of layers gives the depth of the neural network model. Based on this, output *y* of the sample NN for inputs $\{x_1, x_{2,\ldots}, x_n\}$ is computed as the composition of application of functions of particular layers in feed forward way, e.g.:

$$y = f^{(3)}(f^2(f^{(1)}(x))) \tag{3}$$

3.2 Backpropagation Algorithm

Backpropagation algorithm (BP) is a gradient-based algorithm for learning neural networks using the training set $\{(x^{(1)}, y^{(1)}), \ldots, x^{(m)}, y^{(m)})\}$ of *m* examples. The goal of the BP procedure is to minimize cost (error) function, which represents the difference between the expected output for a training example and the real output of the network. While there are different types of cost function, for BP procedure explanation let's assume we applied the mean-square cost function:

$$J(t) = 0.5 \sum_{i=1}^{N_o} (eo_i(t) - y_i(t))^2 \tag{4}$$

where *t* is an index of *t*-th training example, N_o is a number of neurons in the output layer, and *eo* is a real (expected) output value from the training set. An optimization process is based on the modification of weights of the neuron for time *t*+1 from previous values in time *t* as follows:

$$w_{ij}(t+1) = w_{ij}(t) - \Delta w_{ij}(t), \Delta w_{ij}(t) = \alpha \frac{\partial}{\partial w_{ij}} J(t) \tag{5}$$

where α is the learning rate. The computation of Δw_{ij} can be written as:

$$\Delta w_{ij}(t) = \alpha \frac{\partial J(t)}{\partial in_i(t)} \frac{\partial in_i(t)}{\partial w_{ij}(t)}, \frac{\partial J(t)}{\partial in_i(t)} = \delta_i(t), \frac{\partial in_i(t)}{\partial w_{ij}(t)} = x_j(t) \tag{6}$$

then weight update can be written as:

$$\Delta w_{ij}(t) = \alpha \delta_i(t) x_j(t). \tag{7}$$

Now the problem of NN learning is to find $\delta_i(t)$ for every neuron in the network. This leads to a simple recursive equation for $\delta_i(t)$ computation (for *i*-th neuron in the layer), which represents backpropagation of error.

For output layer neurons equation for $\delta_i(t)$ is:

$$\delta_i(t) = \big(eo_i(t) - y_i(t)\big) f'(in_i(t)) \tag{8}$$

and for neurons in the hidden layer $\delta_i(t)$ is computed as (where N_o is number of neurons on the output layer to the current hidden layer neuron):

$$\delta_i(t) = f'(in_i(t)) \sum_{h=1}^{N_o} \delta_h(t) w_{hi}(t) \tag{9}$$

While backpropagation can be applied in the same way (or with some changes related to faster and more effective learning), the main difference of the expected output model of NN and its application can be achieved by the use of a different cost function.

4 Our Topic Modeling Autoencoder

An autoencoder is a type of neural network with a hidden layer, which is trained to provide the same output on its output layer as input on the input layer. Then, the hidden layer (with a smaller number of neurons) holds encoding information on training examples. Usually, autoencoder is used for dimensionality reduction or features learning. Simply, autoencoder network can be viewed as a two-part network with: the encoder function *h = f(x)* and decoder function *r = g(h).* The encoder is used to reduce dimensionality and produce a smaller number of input features, while the decoder is used to reconstruct original input from reduced representation in the hidden layer. The basic architecture of the autoencoder is shown in Figure **6**. In this case, we have layers $X \approx R$ and Z is a representation (coding) of inputs with reduced dimensionality.

4.1 Autoencoder for Topic Modeling

Our topic modeling autoencoder (TopicAE) is inspired by the sparse autoencoder presented in [17]. The architecture of the autoencoder is the same as in Figure **6**, where input layer *X*is used for documents represented by words (containing

variables x_i for every word in documents within a corpus) and hidden layer Z providing induced topics in its K neurons (with k-thtopic represented by z_k).

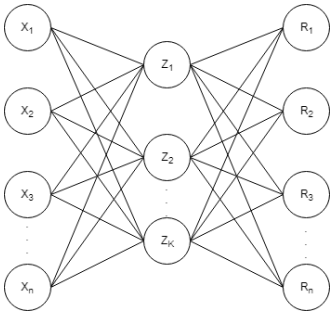

Figure 6

The architecture of autoencoder with n input/output neurons (with particular variables within input layer X and output layer R) and K neurons in a hidden layer. In the case of TopicAE, variables in layer X represent particular words in the vocabulary of documents in the corpus and Z contains K different topics.

For TopicAE, we propose a topic penalty on encoder (hidden) layer units, which is based on a sparse penalty. Moreover, we need to achieve generative properties of the method to provide an approach which imitates the behavior of generative topic models such as LDA. Such assumption leads to the approach where only several neurons in the hidden layer (representing topics) should be activated (neuron is activated when its value is near 1 and inactive when its value is close to 0) for each input (document), and also each topic should be activated only for several documents. To achieve this behavior, we added a topic penalty to cost function in the hidden neurons layer as described in the following equations:

$$J_{topic}(t) = J(t) + \Omega(t) \tag{10}$$

$$\Omega(t) = \alpha \sum_{i=1}^{m} KL(\rho||\rho_i') + \beta \sum_{i=1}^{m} KL(\varsigma||\varsigma_i') + \gamma \sum_{i=1}^{h} KL(\sigma||\sigma_i') \tag{11}$$

where α, β, γ controls the weight of the penalty terms in cost function (usually set to 1), m is a number of training examples, h is a number of neurons in the hidden layer. $KL(\rho||\rho_i')$ represents the penalty based on KL divergence (which describes the divergence of one probability distribution to another) chosen as follows:

$$KL(\rho||\rho_i') = \rho \, log \frac{\rho}{\rho_i'} + (1-\rho) log \frac{1-\rho}{1-\rho_i'} \tag{12}$$

where ρ_i' is average activation of the hidden units for i-th training example, ς_i' is median of activations of the hidden units for i-th training example, σ_i' is average activation of the hidden units over the training set and $\rho, \varsigma, \sigma$are constants (typically small values close to zero, e.g.,0.05 - to make average values of ρ_i',ς_i',σ_i' of each hidden neuron to be close to 0.05). To achieve similar distribution of topics for each training example (document) as in classical topic modeling methods, $\varsigma<\rho$ should be true.

In TopicAE, we represent each input text document as a log-normalized word count vector $\boldsymbol{x} \in R^d$, where each dimension is represented as:

$$x_i = \frac{log(1+n_i)}{max_{i \in V} log(1+n_i)}, \forall\, i \in V \tag{13}$$

where V is the vocabulary and n_i is the word count in the document for i-th word in the vocabulary.

For activation function in each layer we used sigmoid function, and as a cost function, we selected binary cross entropy:

$$C = -\frac{1}{n}\sum_{i=1}^{m}(eo_i\, ln\, y_i + (1 - eo_i)\, ln(1 - y_i)) \tag{14}$$

As mentioned above, topics for each input document can be obtained from the hidden layer, where each neuron represents one of the topics. To find specific words which describe the k-th topic we need to strongly activate particular k-th neuron (set its value to 1 and values of other neurons to 0), compute the output activations and obtain the words which correspond to output units.

In order to see the evolution of topics in time (from data streams or from the whole dataset with timestamps), TopicAE can be simply applied on a chronologically ordered input corpus with the usage of particular documents or their batches (smaller sets of documents from the defined period). The application of TopicAE in batches leads to the visualization of topics evolution in time, where it is possible to follow any topic and changes in its particular description (example of one topic evolution in time is shown in Table 3).

4.2 Extension for Hierarchical Topic Modeling

One of harder tasks in topic modeling is a possibility to extract a hierarchical structure of topics. To extract the hierarchical structure of topics we need to extend architecture and combine more autoencoders in a specific way. It means that for a hierarchical model of topics with depth h, we need to combine h TopicAE autoencoder layers. In practice, such architecture for $h=3$ is shown in Figure 7. Here we have three TopicAE hidden layers to learn three levels of hierarchy topics. Hence, our model is going to learn reconstruction function of input for every possible output $(X \approx R_1, X \approx R_2, X \approx R_3)$. Similarly as in TopicAE case, H_1, H_2, H_3 (where H_1 represents the most specific topics and H_3 the most general topics) represent topics distribution for input and weights between these layers represent dependencies between topics in each layer.

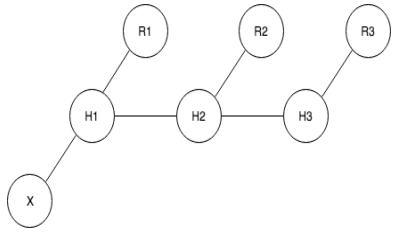

Figure 7
The architecture of the composition of three autoencoders (h=3) for extraction of the hierarchical topic model with three levels of subtopics

The main problem with learning such a composition of autoencoders is that using only J_{topic} penalty, we find it problematic to extract meaningful representation of dependencies between the topic layers. It is simply because TopicAE in the basic setup extracts topics dependencies across layers with similar weights. This behavior is expected if we do not need to have subtopics of some higher topics. On the other hand, in the hierarchical model we expect that whenever a topic is activated on a higher level, on a lower level only subtopics related to such activated parent topic are also activated. This is similar to J_{topic} penalty, but instead of relations topic-document (in simple TopicAE) we need relation topic-subtopic, which leads us to weights between autoencoders' hidden layers. In our composition of TopicAE autoencoders, this problem can be solved with a penalty added to weights between two topic layers, which will add such behavior to a composition. Therefore, in order to achieve this behavior we propose dependency penalty for weights between topics layers as follows:

$$J_{depencency}(t) = J(t) + \Omega(t) \tag{15}$$

and

$$\Omega(t) = \alpha \sum_{i=1}^{s1} KL(\rho||\rho_i') + \beta \sum_{i=1}^{s1} KL(\varsigma||\varsigma_i') + \gamma \sum_{i=1}^{s2} KL(\sigma||\sigma_i') \tag{16}$$

Where*s 1* is a number of weights at a more specific topic layer and *s2* a number of topics in a more general layer. The application of dependency penalty will learn weights between topic layers in the composition of TopicAEs, i.e., this network will activate only some topics in a more specific level (e.g., H_1) that belong to activated topics in a more general level (e.g., H_2). Then, it is only on our decision how many levels of subtopics we want to have and learn the composition of TopicAE autoencoders.

5 Experiments

In this section, we evaluate our proposed TopicAE with other topic modeling methods and its effectiveness in the extraction of topic structure in time as well as the hierarchical structure of topics.

The evaluation was performed on the Reuters Dataset[1], which contains 90 classes, 10788 documents and vocabulary consisting of 35247 unique words, and also on the 20Newsgroups dataset[2], which contains 18846 documents divided into 20 classes. For all of the evaluated methods we preprocessed datasets in the following way:

- Tokenization - split of texts into tokens (in our case words),
- Removing stopwords and words with a length smaller than three characters,
- Word lemmatization and normalization to lowercase form,
- Selection of words which occurred in at least ten documents, but in less than 50% of documents.

The preprocessing steps reduced the initial vocabulary of Reuters dataset from 35247 words to 4672 words. For 20Newsgroups dataset, we took 2000 most frequent words filtered by preprocessing.

For the evaluation of our experiments, we decided to select two standard evaluation metrics. First, we used UMass topic coherence [18] to evaluate the quality of the extracted topics. It represents the pairwise score of *n* top words of the topic and is defined as follows:

$$coherence = \sum_{i<j} log \frac{D(w_i,w_j)+1}{D(w_i)} \tag{17}$$

where $D(w_i)$ is defined as a number of documents containing the word w_i and $D(w_i, w_j)$ is a number of documents containing both words w_i and w_j.

As a second metric, we applied normalized mutual information (NMI) [19], which evaluates how diverse the topics are. For NMI we at first selected top N words (in our case N = 100) for each topic and divided them into 10 clusters $\{<w_1:w_{10}>,<w_{11}:w_{20}>,...,<w_{91}:w_{100}>\}$. Then, these clusters were evaluated on how similar two topics were according to N top words. The final value of the NMI evaluation metric was the average of NMI between every two topics. We also compared our proposed model with several neural network models (ProdLDA, NVDM) and LDA model variations described in paper [26]. For that purpose we also evaluated our model using normalized point wise mutual information (NPMI) topic coherence [26]:

$$NPMI(w_i) = \sum_{j}^{N-1} \frac{log\frac{P(w_i,w_j)}{P(w_i)P(,w_j)}}{-logP(w_i,w_j)} \tag{18}$$

[1]https://martin-thoma.com/nlp-reuters/
[2]http://qwone.com/~jason/20Newsgroups/

5.1 Standard Topic Modeling

Now, we describe the results from experiments in a standard topic modeling task, i.e., we evaluate the quality of extracted topics by TopicAE with other standard methods such as LDA and LSI. For these experiments, we ran TopicAE using autoencoder architecture with one hidden layer and the following parameters for the topic penalty: $\rho = 0.03, \varsigma = 0.01$, $\sigma = 0.03$. The learning phase consisted of 30 epochs. The values of parameters and number of epochs were selected by testing several settings of their values and we selected values which gave us better results.

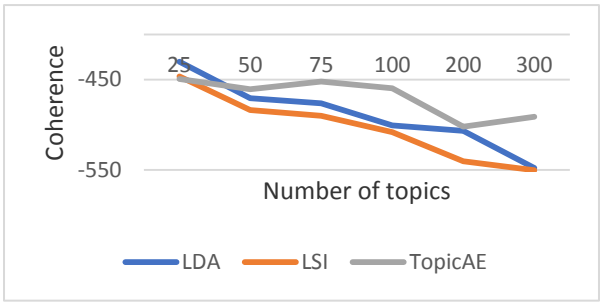

Figure 8
Average topic coherence for LDA, LSI, and TopicAE - Reuters dataset (higher value is better)

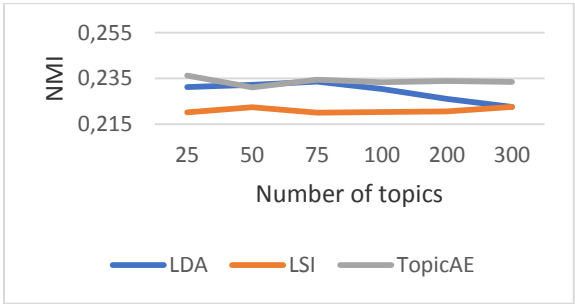

Figure 9
Comparison of average NMI for LDA, LSI, andTopicAE- Reuters dataset (lower value is better)

Figure 8 presents the results of average topic coherence for the compared methods using different numbers of topics (on Reuters dataset). As we can see from the graph, TopicAE out performs standard topic modeling methods according to coherence evaluation metric. The comparison of these methods using NMI (see **Figure 9**) shows that our proposed approach is more likely to generate topics with similar word collocations. Some of the extracted topics are illustrated in Table 1, where we show ad-hoc selected topics from 100 extracted topics for the whole dataset.

Table 1
Example of generated topics by TopicAE

Dollar	Cooper	Orange	Repurchase	Cable
Miyazawa	Ounce	Barley	Corp	Telecommunication
Dealer	Gold	Maize	Reserve	Wireless
Tokyo	Mine	Juice	Customer	Merge
Japan	Loss	Tonne	Temporary	Hold
Tonne	Mining	Argentine	Security	Share
Nakasone	Ton	Gallon	Tonne	Settlement
Stock	Quabec	Grain	Well	Company

Table 2 gives comparison of TopicAE with other topic models (described in paper [26]). From the results, it is obvious that our method out perform other methods by given metric, except ProdLDA.

Table 2
Average NPMI topic coherence on the 20 newsgroups dataset (higher value is better)

# topics	ProdLDA VAE	LDA VAE	LDA DMFVI	LDA Collapsed Gibbs	NVDM	TopicAE
50	0.24	0.11	0.11	0.17	0.08	0.18
200	0.19	0.11	0.06	0.14	0.06	0.17

5.2 Online Topic Evolution in Time

In this subsection, we show how it is also possible to use TopicAE to learn topics evolution in time and evaluate its quality against LDA and LSI. For the purpose of these experiments, we ran TopicAE using the same architecture as in the previous subsection with the same parameters of topic penalty. In this case we only did experiments with Reuters dataset because of time metadata for particular documents. To be able to simulate topic evolution we ordered documents chronologically and divided them into 10 batches $t=\{t_1,t_2,....,t_{10}\}$, each consisting of about 1050 documents.

TopicAE was first learned in the batch with the oldest documents (to be able to learn initial representation of topics we ran TopicAE using 50 iterations). Next, we used the learned TopicAE inthe next batch, and so on. After obtaining the initial topics representation from the first batch run, for all non-initial batches we only used 20 epochs for learning.

In Figure **10** we can see coherence for topics extracted by different models for each of the batches. TopicAE retains stable values of coherence during topics learning and outperforms LDA and LSI. Figure **11** shows that also NMI score has stable values and is still comparable to other topic modeling methods.

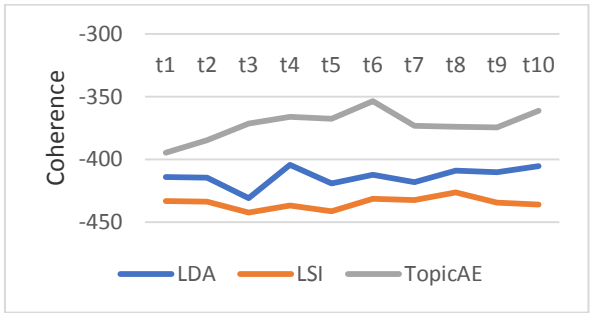

Figure 10
Comparison of coherence for learning topics evolution in time on Reuters datasest batches (higher value is better)

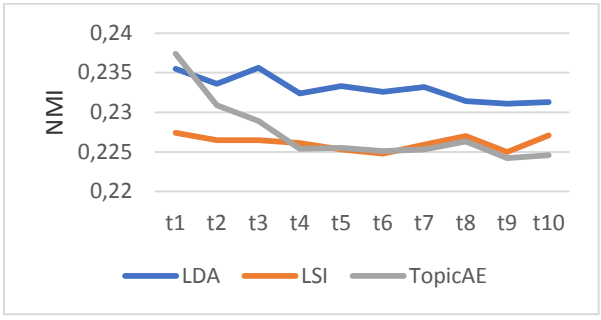

Figure 11
Comparison of NMI for learning topics evolution in time on Reuters datasest batches (lower value is better)

In Table 3 we illustrate an example of topic evolution over the time for one topic extracted using TopicAE. Here, we can see changes in the most characteristic words extracted for the topic during batches from T1 to T10 (words with higher probability are always top within the time batch).

Table 3
An illustrative example of the evolution of topic extracted by TopicAE

T1	T2	T3	T4	T5
Billion	Billion	Week	Week	Week
Week	Dlrs	Billion	Say	Say
Barrel	Week	Dlrs	Dlrs	Money
Rose	Rose	Rose	Barrel	March
Bank	Stock	Barrel	Money	Dlrs
Year	Barrel	Say	Supply	Supply
Stock	Fell	Fell	Rose	Ended
Crude	Supply	Supply	Bond	Growth
Fell	Year	Dollar	Stock	Barell
december	money	government	billion	Rose

T6	T7	T8	T9	T10
Week	Week	Week	Week	Week
March	Ended	Barrel	barrel	Say
Ended	Barrel	Distillate	Distillate	Barrel
Demand	March	Gasoline	Gasoline	Gasoline
Say	Distillate	Weekly	Weekly	Distillate
Crude	Gasoline	Demand	Demand	Stock
Economic	Crude	Stock	Stock	Weekly
Stock	Stock	Ended	Ended	Demand
Distillate	Say	Say	Say	Ended
supply	petroleum	crude	crude	crude

5.3 Hierarchical TopicAE

The composition of TopicAE autoencoders is also applicable for the extraction of the hierarchical structure of topics. For the purpose of these experiments we used TopicAE layered architecture with two hidden layers, one with 200 neurons (more specific layer) and second with 50 neurons in hidden layers (more general layer), and with the following parameters for each of the topics and dependency penalties: $\rho = 0.05, \varsigma = 0.03,\ \sigma = 0.05$. We used 30 epochs for learning on 20Newsgroups dataset, which was preffered for experiments with hierarchical structuring due to better separation of main classes. Similar to previous experiments for simple topic modeling, the parameters and number epochs were selected by previous testing of several settings.

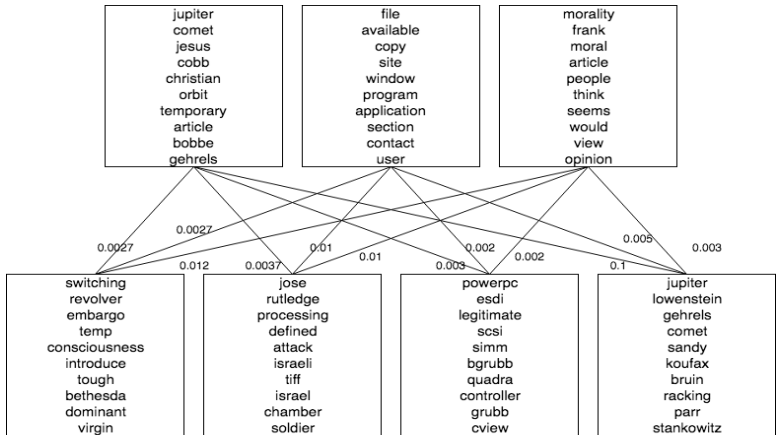

Figure 12

Part of the extracted hierarchical structure from the 20Newsgroup dataset (TopicAE with two levels)

In general, there is currently nohierarchy-based evaluation metrics available. Therefore, we applied evaluation metrics in a particular hierarchical level. Here, we achieved *coherence = -581.3* and *NMI = 0.25* for topics in a more specific TopicAE level (layer with 200 neurons). In a more general TopicAE level (with

50 neurons in the hidden layer), evaluation metrics had values of *coherence* = *-458.6.97* and *NMI* = *0.235*. Figure **12** shows part of the extracted hierarchical structure. Here we can see that hierarchical TopicAE architecture was able to discover meaningful topics on each hierarchy level and also to find similar topics across the hierarchy.

Conclusion

In this paper, we proposeda novel neural network approach to solving the topic modeling problem usingTopicAE autoencoder. The main advantage of this solution is that it can be applied to classical topic modeling, topic modeling over time and to the extraction of the hierarchical structure of topics. Our experiments showed that TopicAE could extract topics with similar or better quality (measured by evaluation metrics) than other standardly applied models. Also, our model was likely to generate more topics with similar words.

For future work, we want to eliminate the problem of most topic models, which is that they are only able to work with a bag-of-words model for their input. We expect that it can be solved using additional layers in our network, i.e. it will be possible to represent the input using an embedding layer in our network.

Acknowledgment

The work presented in this paper was supported by the Slovak VEGA research grant 1/0493/16,and Slovak APVV research grants APVV-16-0213 and APVV-17-0267.

References

[1] T. K. Landauer, P. W. Foltz and D. Laham: An introduction to latent semantic analysis, *Discourse Process*, Vol. 25, pp. 259-284, 1998

[2] D. M. Blei, A. Y. Ng and M. I. Jordan: Latent dirichlet allocation, *JMLR*, Vol. 3, pp. 993-1022, 2003

[3] T. Hofmann: Probabilistic latent semantic indexing, *SIGIR99*, 1999, pp. 50-57

[4] J. Petterson *et al.*: Word features for latent dirichlet allocation, *NIPS*, 2010, pp. 1921-1929

[5] K. Zhai and J. Boyd-Graber: Online Latent Dirichlet Allocation with Infinite Vocabulary, *ICML*, 2013, pp. 561-569

[6] Y. W. Teh*et al.*: Sharing Clusters among Related Groups: Hierarchical Dirichlet Processes, *NIPS*, 2004, pp. 1385-1392

[7] D. M. Blei and J. D. Lafferty: Dynamic topic models,*ICML*, 2006, pp. 113-120

[8] Ch. Wang, D. Blei and D. Heckerman: Continuous time dynamic topic models, *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, 2012, pp. 579-586

[9] A. Beykikhoshk *et al.*: Discovering topic structures of a temporally evolving document corpus, *KAIS*, Vol. 55, pp. 599-632, 2018

[10] D. M. Blei, T. L. Griffiths and M. I. Jordan: The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies, *JACM*, Vol. 57, 2010

[11] T. Hoffman: The cluster-abstraction model: Unsupervised learning of topic hierarchies from text data, *IJCAI*, Vol. 99, 1999

[12] A. Smith, T. Hawes and M. Myers: Hierarchie: Interactive Visualization for Hierarchical Topic Models, *ILLVI*, 2014, pp. 71-78

[13] W. Li and A. McCallum: Pachinko allocation: Scalable mixture models of topic correlations, *JMLR*, 2008, Submitted

[14] P. Chen *et al.*: Progressive EM for Latent Tree Models and Hierarchical Topic Detection, *AAAI*, 2016, pp. 1498-1504

[15] J. Paisley *et al.*: Nested hierarchical Dirichlet processes, *TPAMI*, Vol. 37, pp. 256-270, 2015

[16] *Deep Learning*, I. Godfellow, Y. Bengio and A. Courville, MIT Press, 2016 [Online] Available: http://www.deeplearningbook.org/

[17] A. Ng: Sparse autoencoder, CS294A Lecture Notes, Vol. 72, pp.1-19, 2011

[18] D. Mimno *et al.*: Optimizing semantic coherence in topic models, *Proceeding of EMNLP* 2011, pp. 262-272, 2011

[19] A. F. McDaid *et al.*: Normalized Mutual Informationtoevaluate overlapping community finding algorithms, 2011

[20] R. Salakhutdinov and G. Hinton: Replicated Replicated Softmax: an Undirected Topic Model, *NIPS*, 2009, pp. 1607-1614

[21] H. Larochelle and S. Lauly: A Neural Autoregressive Topic Model, *NIPS*, 2012, pp. 2708-2716

[22] Y. Miao, L. Yu and P. Blunsom: Neural Variational Inference for Text Processing, *ICML*, 2016

[23] Z. Cao*et al.*: A Novel Neural Topic Model and Its Supervised Extension, *AAAI*, 2015, pp. 2210-2216

[24] Y, Chen and M. Zaki: KATE: K-Competitive Autoencoder for Text, *KDD*, 2017, pp. 85-94

[25] P. Gupta *et al.*: Deep Temporal-Recurrent-Replicated-Softmax for Topic Trends over Time, *NACCL HLT*, 2018, pp. 1079-1089

[26] A. Srivastava and Ch. Sutton: Autoencoding Variational Inference For Topic Models, *ICLR*, 2017