An Autoencoder Approach to Topicmodeling

Simon Roth
s.roth@university-konstanz.de
Student ID: 01950332

University of Konstanz
Computational Social Science
Dr. Karsten Donnay
Winter Term 2017/2018

Abstract: This paper aims to explore latent class models and their application to texts from social sciences. Latent Dirichlet Allocation (LDA) is a popular model to compress texts into topic vectors in an unsupervised fashion. More recently autoencoders (AE) have been proposed to tackle the problem of high dimensionality of text with the power of deep learning. Variational autoencoder (VAE) brings the world of Bayesian inference to high capacity neural networks. As the experimental part shows, VAE systematically outperforms LDA on several information retrieval tasks and establishes visually the most information-rich latent space of all benchmarked models. High quality document encodings are able to accuartly detect toxic language, fake news or subjects of scientific publications.

1 Introduction

This paper aims to explore latent class models and their application to texts from social sciences. The most popular model for compressing text into a set of latent variables is Latent Dirichlet Allocation (LDA). More recently autoencoders (AE) have been proposed to tackle the problem of high dimensionality of text with the power of deep learning. Beside LDA, this paper will include two types of autoencoders, namely sparse (SAE) and variational (VAE). In order to grasp the unsupervised learning behavior of these models a range of experiments have been conducted on three different datasets. Benchmarking these models allows to determine the most useful one in terms of visual discovery and predictive capabilities. For example the obtained document vectors form LDA, SAE, and VAE can be further used to build on top supervised classification algorithms that are trained with a gold standard outcome.

Autoencoders have been developed only in the past five to ten years which provide a new alternative to latent class models with high dimensional sparse input. Like the majority of deep learning concepts in natural language processing (NLP), autoencoders were not invented for the propose of text input, but for images. Nonetheless they have shown great performance an a variety of NLP tasks like document classification or retrieval. New tools have been developed all over the place to handle long standing NLP problems like question-answering, entity recognition and language generation. All these innovations are driven by neural networks and their capability of non-linear feature fusion. That means a network can learn over several layers highly conditional and hierarchical word associations. Moreover, neural network approaches to topicmodelling have some other benefits to traditional ones. While traditional machine learning models like LDA scale linearly with the size of the training data, neural networks only scale linearly with the number of weights in a network, regardless of input size (Goldberg 2016: 371). This is probably the most important argument for researchers in a big data environment.

The rest of the paper is organized as follows. In Section 2, all model architectures are introduced beginning with LDA and the posterior distribution as an exploratory tool for large corpora. Furthermore, the basics of autoencoders and their general placement in the neural network community are discussed. The most promising model, a variational autoencoders (VAE) will combine the best of two worlds: A neural network architecture with a variational Bayesian approach to latent encoding. Therefore this VAE is expected to outperform all other models in the subsequent experiments in section 3.

2 Models

2.1 LDA

Latent Dirichlet Allocation (LDA) is a mixture model applied to the textual domain. In fact it is a hierarchical latent mixture model estimated with Bayesian inference on a document term matrix. Formally, they belong to the hidden or latent variable models (Blei & Lafferty 2009: 3). Latent variable models are structured distributions in which observed data interact with hidden random variables. Furthermore, LDA is a generative model which attempts to discover the latent semantic structure in a collection of documents automatically without supervision (target/ outcome). The emerging document/topic vectors θ as well as the related word topic assignments β are latent variables. LDA works on a distribution over a fixed vocabulary (number of unique words) and fixed number of topics k. In each topic the distribution of words are different. The documents are then assumed to be generated by the following process:

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) = \prod_{i=1}^{K} p(\beta_i) \prod_{d=1}^{D} p(\theta_d)(\prod_{n=1}^{N} p(z_{d,n}|\theta_d)p(w_{d,n}|\beta_{1:K}, z_{d,n}))$$

In order to allow the mixing proportions to be document specific, it is usual to assume a distribution over the mixing proportions (Bayesian perspective, distributions of distributions), although the mixing proportions could be seen as random effects. Figure 1a shows the graphical notation of LDA and plot b the enrolled hierarchical model/tree of Bayesian topicmodels. The posterior distribution on the mixing proportions can be viewed as the representation of a document in the context of a given corpus. By applying Bayes rule, the computational task of inferring the latent topics is equivalent to computing the following joint probability distribution over the observed and hidden random variables (Blei & Lafferty 2009):

$$p(\beta_{1:K}, \theta_{1_D}, z_{1:D}, w_{1:D}) = \frac{p(\beta_{1:K}, \theta_{1_D}, z_{1:D}, w_{1:D})}{w_{1:D}}$$

Calculating the numerator is straightforward, but computing the normalization factor is extremely difficult for these kind of complex and generalized models. The full set of conditional posterior for all parameters can be computed by summing the joint distribution over every possible instance of hidden topic structure which makes the problem intractable, especially for big data applications. Integrating this multidimensional target function requires a Gibbs sampler or variational inference.

LDA comes with well known costs and limitations (for a detailed report on limitations read: Blei & Lafferty (2009)). Some of these constraints apply to autoencoders as well. First, LDA assumes the estimated topics to be static or stationary over time. A way to relax this assumption is to use a dynamic topic model that explicitly define a temporal dimension. Furthermore, topic models are by design not able to learn correlated topics,

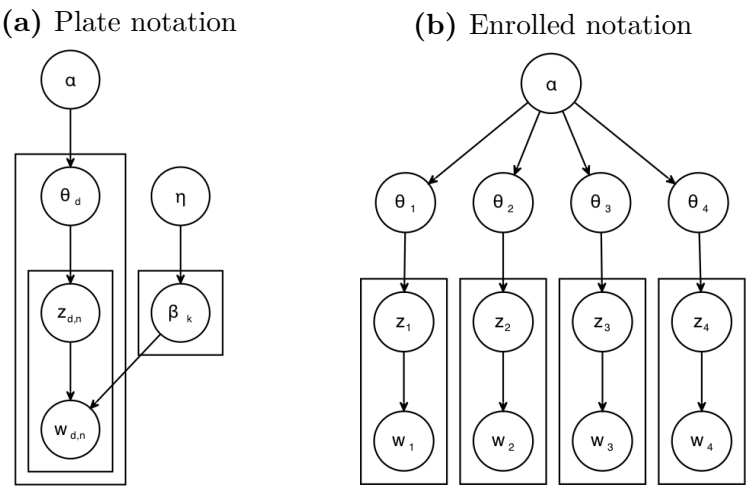

Figure 1: General LDA Structure

which is problematic because such topics commonly occur in the context of the social sciences (CTM can mitigate this issue of vector independence). Indeed, topic models are hierarchically estimated, but they still do not have a notion of abstraction hierarchy, as documents often mix words at different level of abstraction in addition to mixing over topics. One possible solution is to introduce an additional level of hierarchy, which would allow the model to share data between different abstraction levels. Finally, topic models are built upon a document term matrices, a so called bag of words model, which assumes that each word is exchangeable, with no particular order or sentence structure. This procedure loses all local information about which words appear in a similar context. In fact, the following autoencoders are also build on a DTM, therefore losing local structure too, but with capabilities to model the underlying data in highly non-linear fashion.

2.2 Autoencoder

In this section the concepts of autoencoders (AE) are introduced, describing their basic architecture as artificial neural networks (ANN) as well as the activation functions usually applied to their layers. After that, the distinction between sparse (SAE) and variational autoencoder (VAE) will be explained.

An autoencoder is a type of ANN which is used to learn efficient discrete encodings in an unsupervised manner. Applied to text, the goal becomes to learn a lower dimensional feature representation on a document term matrix. Figure 3a shows the basic structure of an autoencoder, which uses an encoder f to reduce w words in the vocabulary to a latent space θ. Afterwards a decoder g network is used to predict each word in the documents again. Therefore, AEs are trained to reconstruct their input to the output layer, while verifying certain restrictions which prevent them from simply copy and pasting (Charte et al. 2018: 3). A shallow AE network for text consists of just one hidden layer, and is defined by two weight matrices and two bias vectors. Whereas the first equation encodes the words into a latent representation z, the second one reconstructs the words given the

latent vectors. The encoder and decoder functions, are two seperate NN, one for data compression one for reconstruction:

$$z = f(z|w) = s_1(W^{(1)}w + b^{(1)})$$
$$r = g(r|z) = s_2(W^{(2)}z + b^{(2)})$$

Typically, autoencoder are symmetrically built neural networks with one, three or five hidden layer, thus belonging either to shallow or deep undercomplete networks (Charte et al. 2018: 5). The central layer has substantially fewer dimensions k, which compress the data into a latent semantic space. This is typically referred to as a "bottleneck", as the encoder must learn an efficient compression of the data into this lower-dimensional space, which can be seen in figure 3b. Typically, feeding data through the network and adjusting weights is done during training phase. After randomly initializing the weights of the network, each is updated using back-propagation algorithm with the goal of minimizing the error of the output. In the case of autoencoder for text, each input is a word column and each output neuron is reconstruction probability for this word.

Each neuron has an activation function that disentangles the factors explaining the variations in the data (Wan et al. 2013). Therefore they determine whether there is enough informative input at a node to fire a signal to the next layer. In figure 5 we see popular activation functions like ReLU or Softmax. First, Rectified Linear Unit (aka Rectifier, ReLU, Max, Ramp Function) has some interesting properties like non-negativity which gives rise to real zero weights but entirely neglects the negative activation space (Glorot et al. 2011). Second, Softmax or Sigmoid functions which are well known as logistic transformations in statistical inference. In fact they are functional equivalents to the log-likelihood estimated by Maximum Likelihood. For classification purpose the sigmoid is often used in the output layer which non-linearly transforms a linear projection into $[0 \leq p \leq 1]$ probability space (Karlik & Olgac 2011). Tanh is also a sigmoid function, but with symmetry at the origin and a steeper slope. Whereas tanh produces supposedly

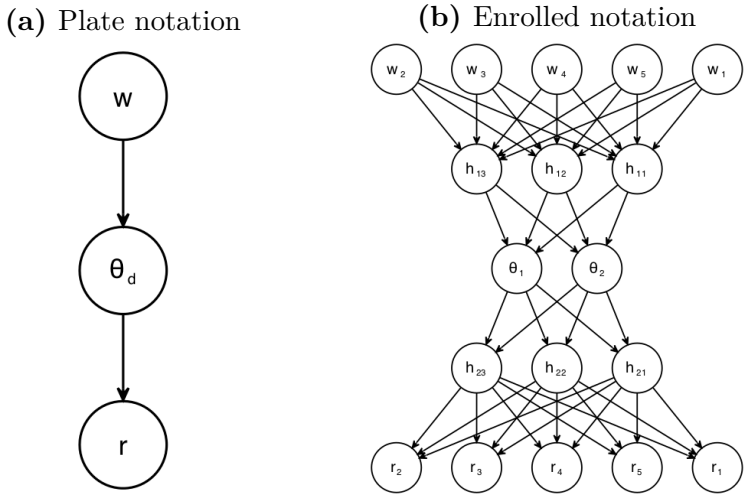

Figure 3: General Autoencoder Structure

stronger gradients, it is possible to design an AE with multiple hidden layers and different activations to combine the properties of several of these functions (Charte et al. 2018: 4).

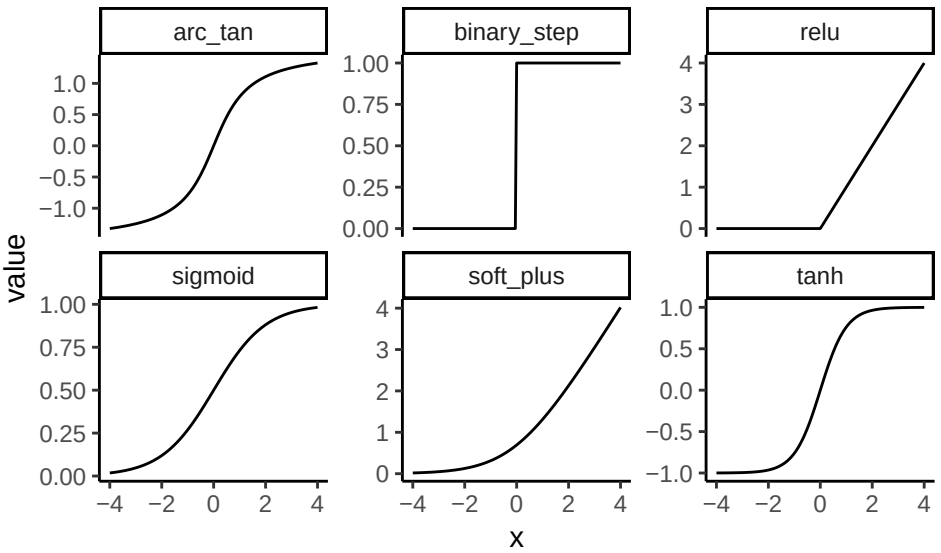

Figure 5: Common Activation Functions for Neural Networks

2.2.1 Sparse AE

Sparse autoencoders are a special flavor autoencoder with a sparsity penalty that forces a single layer network to learn very distinctive features by restricting the number of words required for reconstruction. Due to the massive capabilities of complex networks with several million of weights trained, over-fitting has always been an issue. To alleviate this problem L_1 or L_2 norms can be added to the loss function to perform weight regularization. This will force small weights to be nearly equal to zero and large weights decay by multiplying them with a squared penalty term (UC 2018). In the case of the AE sparsity is introduced by a Bernoulli random variable, assuming a unit can only either fire or not (Charte et al. 2018: 8).

$$\hat{p}_i = \frac{1}{||S||} \Sigma_{x \in S} f_i(x)$$

This kind of constraint forces the algorithm to reduce the input to a single class value that minimizes prediction error. Another way to adaptively regularize a network is to drop out some neurons in order to improve its generalization (Hinton et al. 2012). Dropout randomly forces a set of neurons to be zero and prevents the network from learning to rely on specific weights (Goldberg 2016: 379). This leads to a pruned network which mitigates the risk of over-fitting and can also sparse activations. Typically a drop out ratio of 50% is considered to be optimal for several applications (Srivastava et al. 2014; Zhang & Wallace 2015). Work by Srivastava et al. (2014) establishes a strong connection between dropout and L2 regularization, which applies to this case.

2.2.2 Variational AE

Variational autoencoders (VAEs) were firstly introduced by Kingma & Welling (2013) and Rezende et al. (2014). In fact, VAE only resemble a traditional autoencoder by its architecture as encoder and decoder. At its center layer it takes a variational Bayesian approach to encoding. Its building block are latent, unobserved random variables z, which could have generated the observations x. In comparison to classic autoencoders, VAE replace the deterministic function of the decoder by stochastic mappings that allow to generate high quality vectors. Thus the objective function is to approximate the distribution of the latent variable given the observations (Charte et al. 2018). The objective of VAE in this case has the following form:

$$\mathcal{L}_{VAE}(\theta, \phi; w) = KL(q_\pi(z|w)||p_\theta(z)) - E_{q_\phi(z|w)}[\log p_\theta(r|z)]$$

The loss function consists of two separate terms: first a latent loss (KL divergence) that measures how closely the latent variables match a Gaussian and the second, the generative loss, which is a mean squared error that measures how accurately the network reconstructed the input text. The first term allows the model to learn a sufficiently diverse space and therefore creates information-rich latent variables. In order to optimize the KL divergence via backpropagation a simple reparameterization trick is required: instead of the encoder generating a vector of real values, it will only generate a vector of means and a vector of standard deviations. The decoder networks than samples form a normal Gaussian with the respect to the learned encoder mean and SD. This sampling procedure is well summarized by Doersch (2016) where figure 6 is taken from.

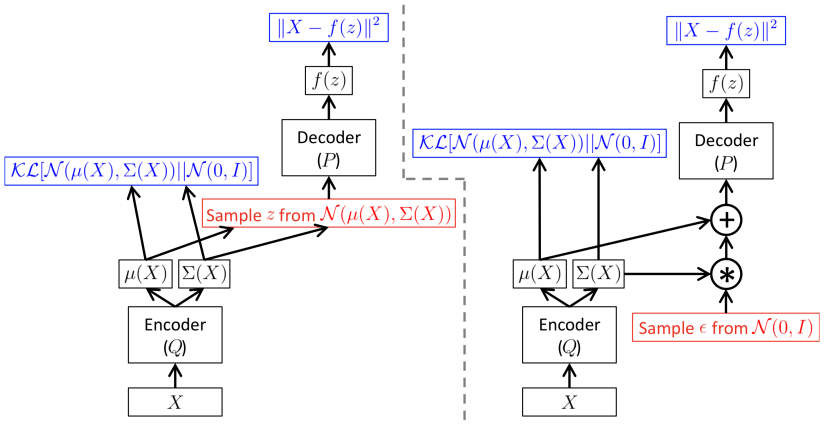

Figure 6: This illustration is taken from Doersch (2016: 10). It shows the same variational autoencoder implemented as a feed-forward neural network, where $P(X|z)$ is a Gaussian, on the left without the "reparameterization trick" and on the right, with it. Sampling operations that are non-differential are shown in red. Blue color denotes loss layers. The feed forward behavior of these networks is identical, but back propagation can be applied only to the right network.

3 Experiments

In this section LDA, SAE and VAE will be evaluated on three different datasets. For evaluation different kind of procedures can be employed to test whether the document vectors of each model actually have learned useful feature representations. All experiments were conducted on a macbook with a 2,3 GHz Intel Quad-Core i5 and 16G RAM. After some testing, the `text2vec` package was used for efficient topicmodeling (Selivanov & Wang 2016) and `ruta` for the autoencoders based on Tensorflow (Charte et al. 2019).

3.1 Data

Recently, Google, Wikipedia and Jigsaw launched a project involving five thousand crowd-workers to annotate approx. 160k comments with toxicity scores in order to train machines to automatically detect online insults, harassment, and abusive speech (Google & Wikipedia 2015). This dataset has been down sampled to counter class imbalance, as only 10% contain toxic commentary, resulting in a total of 28222 instances. The second data set comes from a Kaggle in-class competition. This dataset contains US news articles which are labeled as fake (not reliable) and non-fake. No balancing was required to get 20328 news articles with a label. Lastly, Scopus data was taken from 5 different disciplines namely Medicine (MEDI), Social Sciences (SOCI, Psychology (PSYC), Economy (ECON), and Engineering (ENGI). Each category is almost equally sampled from a much larger corpus due to performance reasons of the following evaluations.

var	fake	pubs	toxic
mean_nwords	764.041	158.705	61.419
num_classes	1	5	1
sparsity	0.994	0.991	0.997
test_size	4066	4662	5644
total_n	20328	23310	28222
train_size	16262	18648	22578

Table 1: Dataset Descriptives and Parameter

Each model presented takes as input a document term matrix with words as columns and documents as rows (Soderland 2001). Therefore, the size of the vocabulary drastically matters in terms of accuracy and speed. Two feature normalization steps have been performed to reduce sparsity. First, lemmatization which removes all word inflictions and second stop words removal. After tokenizing the texts, only the most frequent 10000 words are kept to vectorize the whole corpus. Before training, the data was split into 80% train and 20% test data. Model validation is automatically performed during training. For each combination of data and model type a grid of models has been computed with varying latent dimensions ranging from k = [2, 5, 10, 20, 32, 64, 128, 256]. The size of the latent space is the only common hyperparameter between LDA and AE beside textual parameters.

3.2 Exploration

3.2.1 Word Embedding

The overall object of these algorithms might be to produce document vectors but word vectors can be generated too. In the case of AE, each input neuron (word in the vocabulary) is connected to each hidden neuron with different weights. Usually, models can be visually evaluated by checking whether similar words are closely related to each other in the vector space. Table 2 shows five top words in the word representation space (by cosine similarity) for LDA, SAE, and VAE. A selected number of reference words (column names) are used to retrieve five nearest word neighbors in the latent space. Thereby it can be observed what the different models have learned about various concepts.

data	model	border	law	media	parliament	president	study	weapon
		report	during	policy	country	policy	between	report
		include	change	nation	united	nation	believe	include
fake	lda	recent	company	provide	states	provide	create	recent
		service	accord	student	russia	student	history	service
		likely	public	across	foreign	across	within	likely
		defeat	scalia	rebecca	migrant	before	monkey	vessel
		barack	parish	assail	bloody	fraught	program	exhibit
fake	sae	heroic	paradox	attempt	strain	cellar	common	earlier
		minion	justice	course	unesco	manchin	suffer	fanatic
		illegal	racial	eternal	chamber	island	waiting	corbyn
		deport	commit	network	favour	donald	finding	aurora
		mexico	federal	mirror	austria	peters	harvard	danger
fake	vae	mexican	citizen	outlet	workers	ripple	studies	arsenal
		custom	illegal	behind	britain	during	biology	strikes
		railway	legally	drudge	theresa	united	genetic	lethal

Table 2: Five most similar words in the latent space of fake news articles by cosine similaritys. The column names are the reference words to retrieve 5 similar ones. Thereby we can observe what concepts a model has learned.

3.2.2 Visualization of Document Representations

In figure 7, we observe the PCA projections for each document representation by model and datasets separately. A useful document representation method is expected to cluster related documents, and to differentiate the unrelated ones. Empirically seen, VAE produces the highest quality visualizations that can easily distinguish the given labels without knowing them at training time. In contrast, LDA has a harder time to identify the binary structure of the given clusters. On the Scopus publication data, LDA achieves similar results despite spatial constraints, which prevent the model from enfolding. Figure x,y,z in the appendix present the same latent space but transformed by different dimensionality reduction algorithms (TSNE, UMAP, large Vis).

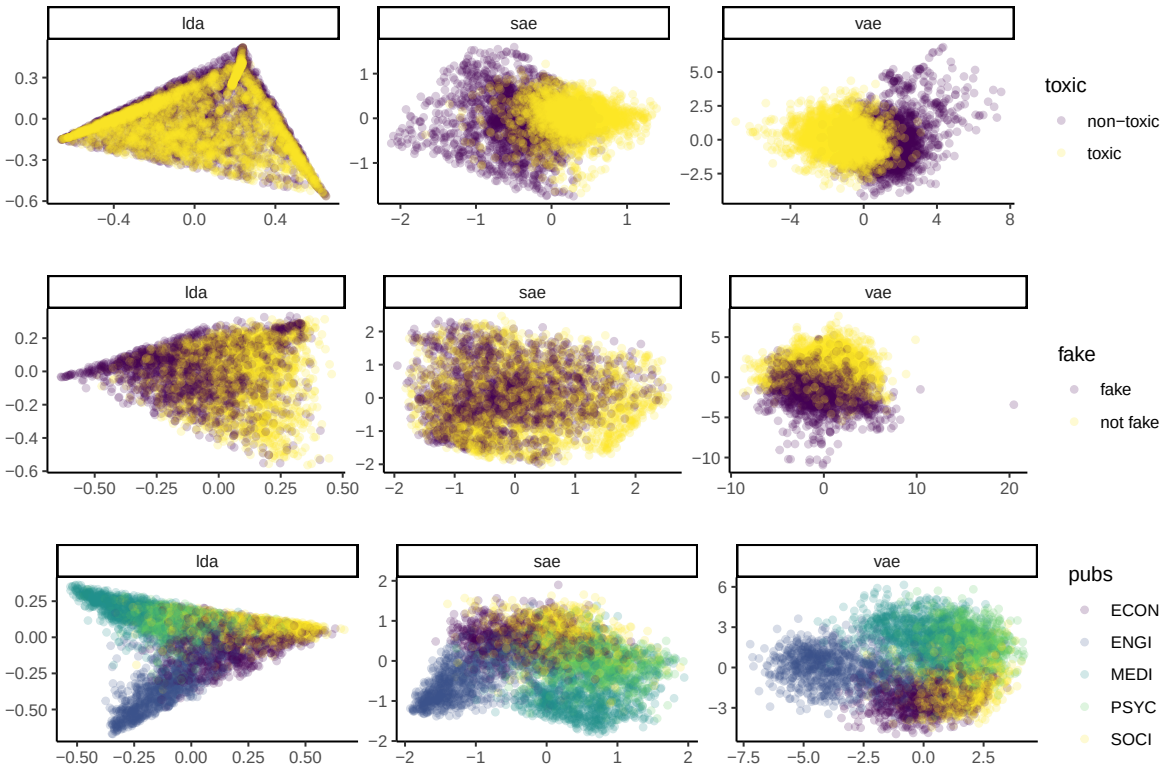

Figure 7: PCA on the 20-dimensional document vectors on various datasets. Each point is a document from the respective test set, which has been encoded/embedded by Latent Dirichlet Allocation (LDA), and sparse (SAE) as well as variational autoencoder (VAE). At training time, the algorithems are unsupervised, meaning they do not know any label or outcome.

3.3 Prediction

This subsection deals with quantitative evaluations of the document representation space by performing classification and document retrieval on it. The chosen datasets come with different labels that can be used to systematically compare the performance of different classifiers trained on top of the document vector space.

3.3.1 Mean Squared Cosine Deviation

As Chen & Zaki (2017) demonstrated, we can use cosine deviation among topics to quantify the distinctiveness of the latent vectors, which can be seen as a measure of how feature rich the newly learned document vectors are. Define the pair-wise mean squared cosine deviation among k topics as follows

$$mscsd = \sqrt{\frac{2}{k(k-1)} \sum cos^2(\theta_i, \theta_j)}$$

Thus, mscd ranges from $[0, 1]$, whereas smaller values indicated more distinctive, respectively orthogonal topics. The next table contains the mscd values for each data set and

model separately, based on k = 20. On this task LDA performs better that SAE but the most distinct vectors are still learned by VAE.

data	lda	sae	vae
fake	0.0402060	0.1525263	0.0108696
pubs	0.0297944	0.2907856	0.0084887
toxic	0.0204477	0.5639036	0.0130606

Table 3: Mean squared cosine deviation among topics; smaller means more distinctive topics.

3.3.2 Document Classification

Now we turn to document classification, the backbone of modern NLP. In this set of experiments two different learners were built upon the document vectors, namely Random Forest (RF) (Wright & Ziegler 2015) and a one layer neural network (NN) (Chollet & others 2015). The NN has either a softmax (multi-class) or sigmoid (binary) as output activation function, depending on the number of levels of the dependent variable. In order to ensure fairness, the same two models are computed for every dataset and document vectors.

For the binary classification task the Receiver Operating Characteristic (ROC) curve is used to assess the accuracy of a predicted probability for predicting a binary outcome. Important for this calculation are two types of errors: the fraction of false positives (TPF), and false negatives (FPF). ROC calculates the cumulative distribution function over a varying discrimination threshold. Usually the cutoff point c is not fixed prior but it can be plotted against TPF and the FPF. Figure 8 shows the ROC curves again for all models and datasets. We can observe that VAE systematically outperforms LDA as well

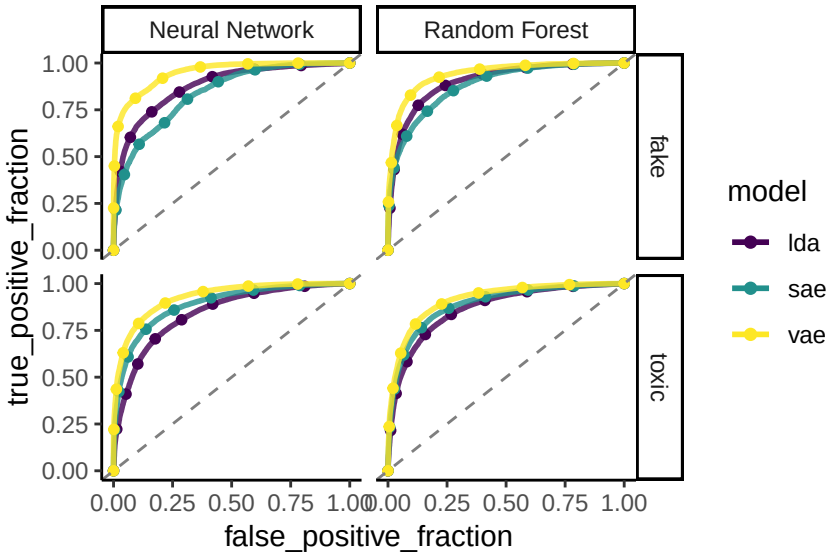

Figure 8: ROC for Binary Classfication

as SAE on the binary classification task, which shows the advantages of VAE over other traditional autoencoders. In table 4 all accuracies are listed with $c = .5$. There is quite a huge discrepancy between the accuracy for the LDA document vectors compared to VAE which is around 10%. Surprisingly, random forest does not learn anything from the toxicity document vectors, which might be caused by short texts (mean words = 61,4).

data	learner	lda	sae	vae
fake	Neural Network	80.6%	84.9%	91.6%
toxic	Neural Network	76.0%	81.0%	84.2%
fake	Random Forest	89.4%	88.5%	92.2%
toxic	Random Forest	49.7%	49.7%	49.7%

Table 4: Classification accuracy (binary data)

The last classification task predicts multi-class outcomes for the Scopus publication data. The following tables show the confusion matrix for each model and learner separately. The best performing model and learner is again VAE in combination with neural networks. This model classifies 8.4 out of 10 texts correctly by only relying on the learned document vectors. The difference of best vs the worst model is again 10% accuracy. LDA seems to systematically confuse some categories compared to VAE.

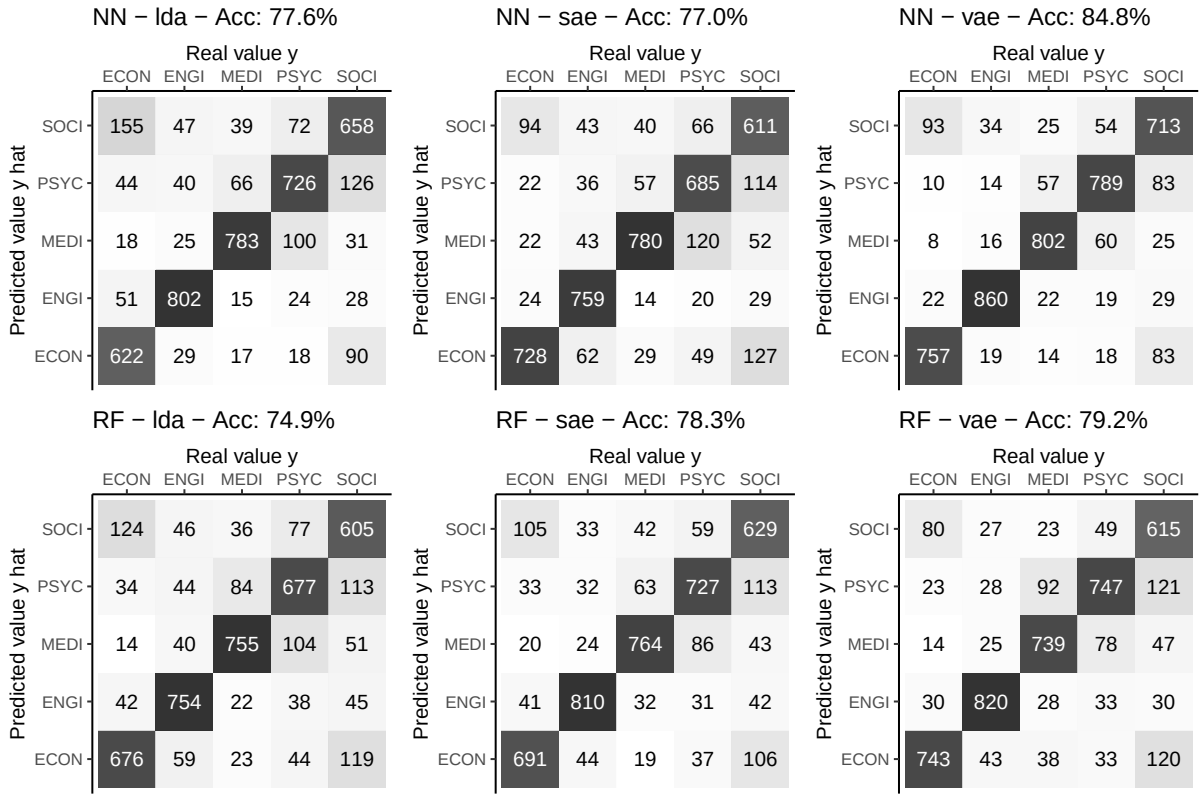

Figure 9: Confusion Matrix for Topic Predictions in Scopus Publication Data

3.3.3 Document Retrieval

The last set of experiments is concerned with document retrieval which is generally defined as the matching of some stated user query against a corpus. For this setup each individual document from the test set is used as query to fetch the relevant documents again from the test set by cosine similarity. Here I follow the example of Chen & Zaki (2017), who used the average fraction of retrieved documents that share the same label as a evaluation metric (precision). This was obviously the computationally most expensive part as on average about 4500 queries with different fractions of the original corpus size had to be computed. This might turn out to be the most useful application of document vectors. Given a database return N similar documents to the input query. Figure 10 shows the a smooth average precision for each model over all document vector sizes [5, 10, 20, 32, 64, 128] (indicated as SE). For the toxicity corpus both autoencoders outperform LDA,

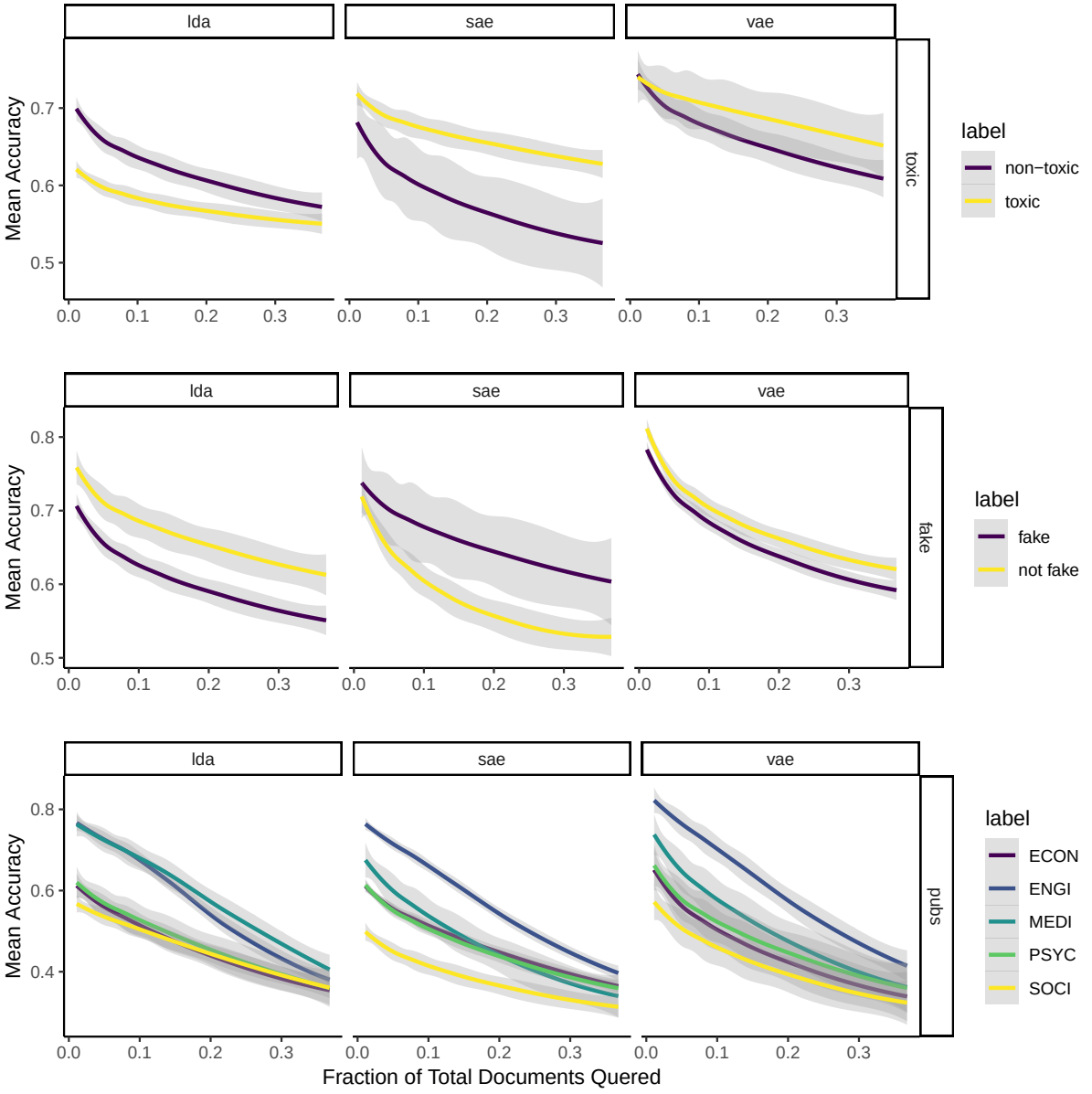

Figure 10: Document Retrieval (Smoothed over all latent topic spaces)

whereas VAE really shines on binary target variables. It is especially worth mentioning, that VAE queries both target labels almost equally well, whereas both other model usually tend towards one good and one bad predicted classes. On the multi-class publication dataset, all models perform reasonably well with some class specific differences. This result encourages to use AE in general to millions of publications, which are in turn ready for user queries from the web.

4 Discussion

This paper demonstrated how different document embedding algorithms can be leveraged in political science practice to detect toxic language, fake news or subjects of scientific publications. After extensive experimentation the most useful model is variational autoencoder (VAE) as this model can build on a neural network architecture and a Bayesian sampling procedure to optimize its stochastic mapping. LDA as well as SAE steadily underperform VAE due to the each models limitations. Whereas VAE tries to maximize the linear separability of classes in the latent space (which take arbitrary values), LDA calculates topic proportions (with arbitrary abstraction level). Despite VAEs predictive power, one might choose LDA as it offers "interpretability" of topics. Recently, KATE a K-competitive Autoencoder for text (Chen & Zaki 2017) was proposed which achieves even higher evaluation scores on sets of experiments outlined in section 3.

There are several avenues to explore. First, I contacted the author of `ruta` to cooperatively add KATE to his package. Second, use skewed distributions to model the posterior of the VAE instead of Gaussian or to implement convolutional or LSTM autoencoder, that are explicitly design to deal with sequential data input like language. The given model could also be applied with one dimension to approximate left-right scale given the underlying text is talking about political issues. After all VAE provide a massive improvement over TF-IDF and LDA methods in terms of predictive power and scalability. Finally, all the intuition build about document vector models and their handling can be directly transferred to organize large collections of scientic literture.

5 Appendix

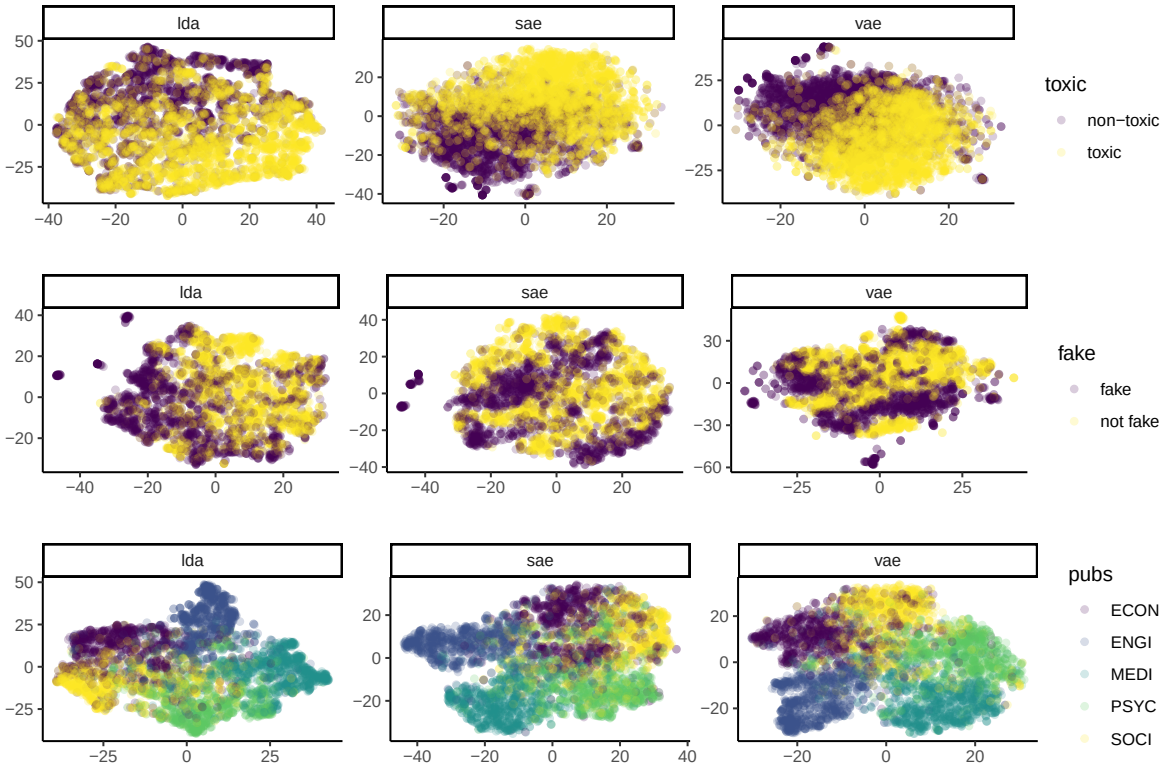

Figure 11: T-SNE on the 20-dimensional document vectors on various datasets

References

Blei, David M & Lafferty, John D 2009: Topic models„ pp. 101–24.

Charte, David et al. 2018: A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines, *Information Fusion* 44, pp. 78–96.

Charte, David, Herrera, Francisco & Charte, Francisco 2019: Ruta: Implementations of neural autoencoders in r, *Knowledge-Based Systems*.

Chen, Yu & Zaki, Mohammed J 2017: Kate: K-competitive autoencoder for text, in *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining*, ACM.

Chollet, François & others 2015: Keras,

Doersch, Carl 2016: Tutorial on variational autoencoders, *arXiv preprint arXiv:1606.05908*.

Glorot, Xavier, Bordes, Antoine & Bengio, Yoshua 2011: Deep sparse rectifier neural networks., 15, pp. 275.

Goldberg, Yoav 2016: A primer on neural network models for natural language processing, *Journal of Artificial Intelligence Research* 57, pp. 345–420.

Google & Wikipedia 2015: Research: Detox, Available at: https://meta.wikimedia.org/wiki/Research:Detox/Data_Release.

Hinton, Geoffrey E et al. 2012: Improving neural networks by preventing co-adaptation of feature detectors, *arXiv preprint arXiv:1207.0580*.

Karlik, Bekir & Olgac, A Vehbi 2011: Performance analysis of various activation functions in generalized mlp architectures of neural networks, *International Journal of Artificial Intelligence and Expert Systems* 1, pp. 111–22.

Kingma, Diederik P & Welling, Max 2013: Auto-encoding variational bayes, *arXiv preprint arXiv:1312.6114*.

Rezende, Danilo Jimenez, Mohamed, Shakir & Wierstra, Daan 2014: Stochastic backpropagation and approximate inference in deep generative models, *arXiv preprint arXiv:1401.4082*.

Selivanov, Dmitriy & Wang, Qing 2016: Text2vec: Modern text mining framework for r, *Computer software manual](R package version 0.4. 0). Retrieved from https://CRAN. R-project. org/package= text2vec*.

Soderland, Stephen 2001: Building a machine learning based text understanding system, in *Proceedings of ijcai workshop on adaptive text extraction and mining*

Srivastava, Nitish et al. 2014: Dropout: A simple way to prevent neural networks from overfitting, *The Journal of Machine Learning Research* 15, pp. 1929–58.

UC 2018: Feedforward deep learning models, *Github Tutorials*.

Wan, Li et al. 2013: Regularization of neural networks using dropconnect, in *Interna-*

tional conference on machine learning

Wright, Marvin N & Ziegler, Andreas 2015: Ranger: A fast implementation of random forests for high dimensional data in c++ and r, *arXiv preprint arXiv:1508.04409*.

Zhang, Ye & Wallace, Byron 2015: A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification, *arXiv preprint arXiv:1510.03820*.