# Learning Topic Models with Arbitrary Loss

Murat Apishev
Moscow State University
Moscow, Russia
great-mel@yandex.ru

Konstantin Vorontsov
Moscow Institute of Physics and Technology
Moscow, Russia
k.v.vorontsov@phystech.edu

*Abstract*—**Topic modeling is an area of text analysis actively developing over the past 20 years. A probabilistic topic model (PTM) finds a set of hidden topics from a collection of text documents. It defines each topic as a probability distribution over words and describes each document as a probability mixture of topic distributions. Learning algorithms for topic models are usually based on Bayesian inference or log-likelihood maximization. In both cases, EM-like algorithms are used. In this paper, we propose to replace the logarithm in the log-likelihood by an arbitrary smooth loss function. We prove that such a modification preserves both the structure of the algorithm and compatibility with any regularizers in terms of additive regularization of topic models (ARTM). Moreover, in the case of a linear loss, the E-step becomes much faster due to the omission of a normalization. We study combinations of the fast and usual E-steps and compare them to regularization using different number of topics in both offline and online versions of EM-algorithm. For an empirical comparison of the algorithms, we estimate perplexity, coherence, and learning time. We use an efficient parallel implementation of the EM-algorithm from the BigARTM open-source library. We show that in most cases the two-stage strategy wins, which uses fast E-steps at the beginning of iterations, then proceeds with usual E-steps.**

## I. Introduction

Topic modeling is a popular technique for understanding the thematic structure of text collections [1]. A probabilistic topic model of a text collection defines each topic by a multinomial distribution over words, and then describes each document with a multinomial distribution over topics. Topical vector representations of texts are widely used in information retrieval, classification, categorization, summarization and segmentation of texts [2]. Historically, the first such model was Probabilistic Latent Semantic Analysis, PLSA [3] and its Bayesian extension named the Latent Dirichlet Allocation, LDA [4]. Since then, topic modelling has been mainly developed within the framework of graphical models and Bayesian learning. Over the past years, hundreds of extensions of LDA and PLSA have been emerged which take into account linguistic considerations, metadata, modalities, languages, topical hierarchies, and various problem-specific requirements [5].

In complicated applications such as exploratory search [6], social media analysis [7], [8], discovering health information from tweets [9], summarizing legislative documents [10], consumer profiling [11], etc. it is desirable for the topic model to satisfy multiple requirements at once. Although the multi-objective nature of topic modeling has been recognized for a long time [12], the popular Bayesian inference techniques did not provide a simple way to combine topic models or to learn them using multi-objective optimization. Additive Regularization of Topic Models (ARTM) is a non-Bayesian approach that addresses this issue [13]. ARTM is not a particular topic model or a family of models, but a general framework for learning combined topic models with desired properties [14].

ARTM is based on the maximization of the log-likelihood criterion together with a weighted sum of auxiliary regularization criteria. Many of topic models originally proposed in Bayesian terms can be re-formulated as likelihood maximization with an additive regularizer [15]. Almost all known types of topic models allow an alternative non-Bayesian representation including multimodal, multilingual, sparse, correlated, temporal, $n$-gram, hierarchical, graph-based, sentence-based, short-text, sentiment and many others. Examples of such a de-Bayesianization can be found in [15]. Then ARTM gives a way to combine topic models by optimizing log-likelihood together with the weighted sum of their corresponding regularization criteria. This technique is called linear scalarization in terms of multi-objective optimization.

This idea led to the modular technology for topic modeling implemented in the BigARTM project [16] — an open-source community-driven library for topic modeling, available at http://bigartm.org. BigARTM implements a fast parallel EM-algorithm leaving the possibility of adding custom regularizers.

The goal of this work is to make the implementation even faster and also reduce memory consumption. To make the EM-algorithm faster, we generalize the log-likelihood criterion replacing the logarithm by an arbitrary smooth monotonically increasing loss function. In the case of a linear loss, this allows to omit the stage of normalization from the E-step of the regularized EM-algorithm. To make the EM-algorithm less memory intensive, we switch to sparse data representation when it becomes profitable.

The paper has the following structure: section II introduces a regularized EM-algorithm for ARTM optimization problem; in section III we infer a new generalized EM-algorithm for arbitrary loss function; section IV describes some implementation issues; section V describes how BigARTM uses model sparseness in order to reduce memory consumption. Quality measures are introduced in section VI; experimental results are reported in section VII; the summary and recommendations are given in section VIII. We conclude in section IX.

## II. Learning topic model with regularization

Consider a set of documents $D$ with a vocabulary of terms $W$. Terms can be words or $n$-grams depending on the text preprocessing used. Let $n_{wd}$ denote the number of times the term $w$ occurs in the document, $n_d$ is the length of the document $d$. *Probabilistic topic model* approximates the observable term frequency in the document $\hat{p}(w \mid d) = \frac{n_{wd}}{n_d}$ by a probabilistic mixture of document-independent term distributions $\varphi_{wt} = p(w|t)$ weighted by conditional topic probabilities $\theta_{td} = p(t|d)$:

$$p(w \mid d) = \sum_{t \in T} p(w \mid t)\, p(t \mid d) = \sum_{t \in T} \varphi_{wt}\theta_{td}. \qquad (1)$$

Learning model parameters $\Phi = (\varphi_{wt})$ and $\Theta = (\theta_{td})$ from data $(n_{dw})$ is a problem of stochastic matrix factorization. This problem is ill-posed, since the set of its solutions is generally infinite. In the additive regularization framework ARTM [13], we find an appropriate stable solution by maximizing the weighted sum of the model log-likelihood and auxiliary regularization criterion $R(\Phi, \Theta)$ under normalization constrains:

$$\sum_{d \in D}\sum_{w \in d} n_{dw} \ln \sum_{t \in T} \varphi_{wt}\theta_{td} + R(\Phi, \Theta) \;\to\; \max_{\Phi,\Theta}; \qquad (2)$$

$$\sum_{w \in W} \varphi_{wt} = 1, \quad \varphi_{wt} \geqslant 0; \qquad \sum_{t \in T} \theta_{td} = 1, \quad \theta_{td} \geqslant 0. \quad (3)$$

Denote by "norm" the operator that normalizes a real vector to make it a discrete probability distribution:

$$p_i = \operatorname*{norm}_{i \in I}(x_i) = \frac{\max\{0, x_i\}}{\sum_{k \in I} \max\{0, x_k\}}, \text{ for all } i \in I;$$

if $x_i \leqslant 0$ for all $i \in I$, then $\operatorname{norm}(x)$ equals zero vector.

The following theorem from [14] gives a simple way to solve the system (2)–(3) numerically for any differentiable $R$. The cornerstone of ARTM approach is that you can learn a huge variety of topic models simply by changing the regularizer $R$ in this general solution. Moreover, you can combine topic models using a weighted sum of their corresponding regularizers instead of $R$. Examples of useful regularizers $R$ can be found in the survey [15].

*Theorem 1:* The local maximum $(\Phi, \Theta)$ of the optimization problem (2)–(3) with differentiable regularizer $R$ satisfies the following system of equations with auxiliary variables $n_{wt}$, $n_{td}$, and $p_{tdw} = p(t \mid d, w)$:

$$p_{tdw} = \operatorname*{norm}_{t \in T}\big(\varphi_{wt}\theta_{td}\big); \qquad (4)$$

$$\varphi_{wt} = \operatorname*{norm}_{w \in W}\left(n_{wt} + \varphi_{wt}\frac{\partial R}{\partial \varphi_{wt}}\right); \; n_{wt} = \sum_{d \in D} n_{dw}p_{tdw}; \quad (5)$$

$$\theta_{td} = \operatorname*{norm}_{t \in T}\left(n_{td} + \theta_{td}\frac{\partial R}{\partial \theta_{td}}\right); \; n_{td} = \sum_{w \in d} n_{dw}p_{tdw}. \quad (6)$$

In the Expectation–Maximization (EM) algorithm, we solve equations (4)–(6) with a fixed-point iteration method, turning these equations into updates. At the start, we initialize $\varphi_{wt}$ by random values, and we assign uniform distributions to $\theta_d$ columns. At each iteration, we compute auxiliary variables $p_{tdw}$ from parameters $\varphi_{wt}, \theta_{td}$ according to *E-step* formula (4), then we compute parameters $\varphi_{wt}, \theta_{td}$ from auxiliary variables $p_{tdw}$ according to *M-step* formulas (5)–(6). We alternate E-step and M-step in a loop until convergence.

PLSA model [3] corresponds to zero regularizer $R = 0$.

LDA model [4] can be considered as a particular case of smoothing (sparsing) regularizer [15] that makes $\varphi_{wt}, \theta_{td}$ values closer to (further from) the given positive (negative) values $\beta_{wt}, \alpha_{td}$:

$$R(\Phi, \Theta) = \sum_{t \in T}\sum_{w \in W} \beta_{wt} \ln \varphi_{wt} + \sum_{d \in D}\sum_{t \in T} \alpha_{td} \ln \theta_{td}.$$

In our experiments we also use the decorrelation regularizer that makes topics as diverse as possible by minimizing the sum of pairwise topic dot products:

$$R(\Phi) = -\frac{\tau}{2}\sum_{t \in T}\sum_{s \in T \setminus t}\sum_{w \in W} \varphi_{wt}\varphi_{ws}.$$

Decorrelation was first introduced in the Topic Weak Correlated LDA (TWC-LDA) model within the Bayesian framework [17] and later adopted as a non-Bayesian regularizer in [14].

## III. Learning topic model with arbitrary loss

Now we generalize the optimization problem (2)–(3), replacing the logarithm in the standard log-likelihood loss $\ln p(w \mid d)$ with a smooth function $\ell$:

$$\sum_{d \in D}\sum_{w \in d} n_{dw} \ell\Big(\sum_{t \in T} \varphi_{wt}\theta_{td}\Big) + R(\Phi, \Theta) \;\to\; \max_{\Phi,\Theta}. \qquad (7)$$

*Theorem 2:* The local maximum $(\Phi, \Theta)$ of the optimization problem (7), (3) with differentiable loss $\ell$ and differentiable regularizer $R$ satisfies the system of M-step equations (5)–(6) and the E-step equation

$$p_{tdw} = \varphi_{wt}\theta_{td}\ell'\Big(\sum_{t \in T} \varphi_{wt}\theta_{td}\Big). \qquad (8)$$

Thus, the system of equations differs from the classical one (4)–(6) only in the E-step equation.

The proof of the theorem can be found in the appendix.

Note that Theorem 1 is an obvious consequence of Theorem 2 for the particular case when $\ell(p) = \ln p$. Only in this case the E-step equation (8) gives the conditional probability $p_{tdw} = p(t \mid d, w)$ in accordance with the Bayes' rule (4).

In the case $\ell(p) = p$, the E-step equation (8) takes the simplest form. Instead of likelihood maximization we maximize the weighted sum of dot-product similarities between model term distributions $p(w|d)$ and their respective empirical estimates $\hat{p}(w|d) = \frac{n_{dw}}{n_d}$:

$$\sum_{d \in D} n_d \big\langle \hat{p}(w|d), p(w|d) \big\rangle + R(\Phi, \Theta) \;\to\; \max_{\Phi,\Theta}.$$

This optimization principle seems as reasonable as the classical likelihood maximization. Moreover, in this case $p_{tdw} = \varphi_{wt}\theta_{td}$. So, we can skip the computation of the normalization factor in (4), which takes a lot of time when processed for each document term. Potentially, this can give a significant acceleration of the EM-algorithm. We call this modification a *fast E-step*.

In the experiments, we will check whether this modification really gives acceleration without affecting the model quality.

## IV. BigARTM LIBRARY

BigARTM is a parallel implementation of EM-algorithm for ARTM model training. The library supports three different data processing modes, namely, offline, synchronous online, and asynchronous online ([18]). All these options were designed to train models on large text collections. So the collection is split into batches of documents for convenient parallel computation. One thread is responsible for processing one batch at a time. BigARTM can fit a model without storing $\Theta$ matrix [16]. This could be achieved by moving computation of per-document parameters $\theta_{td}$ from M-step into E-step. More precisely, by fixing $\Phi$ matrix and proceeding several document passes, on each pass we use 2-step schema:

- compute $p_{tdw}$ variables using (4);
- estimate $\theta_{td}$ parameters with (6).

These steps are repeated till parameters convergence based on some criteria. Then parameters are considered optimal and are used to compute the counters $n_{wt}$. Afterwards both $p_{tdw}$ and $\theta_d$ are removed from the memory. More precisely, to increase the processing speed, the part of the $\Theta$ matrix corresponding to the processed batch is stored in the memory; $p_{tdw}$ are always stored only for the current document.

The offline algorithm proceeds several collection passes, performing the described above E-step for each document. At the end of each pass, we use the final $(n_{wt})$ matrix to construct new $\Phi$ during M-step.

The online algorithm proceeds E-step in the same way, but M-step starts after processing of the given number of batches, instead of the processing the whole collection. While fitting on large collections, it allows us to use the train data for effective $\varphi_{wt}$ parameters update, which leads to faster model convergence. The offline algorithm collects $n_{wt}$ counters during one collection pass and resets them to zeros after M-step. Unlike it, the online one at each moment has two $n_{wt}$ sets: the first counter set is for the current batch, and another one is for all data, processed before. The $(n_{wt})$ matrix, which is used in (5), is estimated by the weighted sum of two sets of counters, thus accumulating all previously processed batches by virtue of exponential moving average.

The asynchronous algorithm also differs from the options described above. Indeed, it updates the $\varphi_{wt}$ parameters in parallel with document processing. It becomes possible due to the presence of two (or even more) sets of $(n_{wt})$ and $\Phi$ matrices. A new set of $\Phi$ based on previously processed documents is prepared simultaneously with the processing of the next set of documents. At this stage algorithm uses a previous version of the parameters. This approach reduces the processing time. The drawback is the increased memory consumption and loss of likelihood at the end of one collection pass compared to synchronous approach under similar conditions. However, the asynchronous algorithm achieves better likelihood than offline or synchronous online do in a given time interval [18].

Note that the presence of a regularizer forces to compute and store the matrix of $r_{wt} = \varphi_{wt}\frac{\partial R}{\partial \varphi_{wt}}$ values at M-step (5). The $(r_{wt})$ matrix has the same size as $\Phi$ and $(n_{wt})$ matrices.

## V. BigARTM OPTIMIZATION FOR SPARSE MODELS

In different ways, the sparsity of topic models is used in many works, for instance, in [19] and [20]. Before we start testing our hypotheses related to the fast E-step, we are going to propose a way to optimize memory consumption and computation time in BigARTM when training sparse models. It allows us to increase the training performance significantly with sparsification regularizers.

BigARTM uses the same data structure to store model parameters $\Phi$, the $n_{wt}$ counters and $r_{wt}$ correction values (let us call it a $\Phi$-like matrix). For now, it is a dense real-valued matrix in which the values are stored line by line, so for each term $w$ the corresponding values are located in memory as a single continuous block. The main advantage is that the memory can be accessed in a locally sequential way while processing loops over a set of topics (as in the 4 formula, which accounts for the bulk of the calculations).

Local access results in significant train speed-up when compared to random access (e.g., in a situation when the sequential loop requires access to different parts of memory). However in case of sparse model calculations for most of the elements will be wasted as they are zero. Primitive solution is to use a condition operator, but it worsens performance even more because multiplication is cheaper than code branching.

Hence we need to process only non-zero elements of $\Phi$ matrix and also save the locality of memory access.

One of the possible solutions is to store $\Phi$-like matrices in CSR (Compressed Sparse Row) format. It reduces the memory required by the model in the case of high sparsity significantly and gives an ability to organize effective loop over non-zero elements in each of its rows. The problem is that in this case, we need to fix the positions of non-zero elements, which is impossible for $(n_{wt})$ matrix, as we collect counters and update its values during E-step. Also, applying regularization may lead to the non-uniform sparse structure of the model, where some parts of the rows will be dense, and some — sparse. In such case, it would be worth using dense representation for dense rows and sparse — for sparse ones.

For these reasons, we propose a hybrid format, in which the storage unit is not the entire matrix, but rather each of its rows separately. For the proposed approach, an array $S$ of length $m$ can be stored in one of two forms, either sparse or dense. The current form choice depends on the number $k$ of non-zero elements in it. In the dense case, the array is stored as a continuous memory block (as it was stored before). The sparse form uses three arrays: $V$, $I$ and $M$. The real-valued array $V$ contains all non-zero elements of $S$ in the original order. An integer array $I$ stores for each element of $V$ its index in the original array. Finally, $M$ is a bitmap with length $m$; its elements are equal to 1 or 0 if corresponding elements in $S$ are greater than zero or equal to it, respectively.

We need $M$ to organize effective ($O(1)$) random access to zero elements of $S$. Indices array $I$ allows $O(\log(k))$ difficulty of access by index operation for non-zero elements of $S$. It also gives ability to proceed a loop over non-zero elements. This feature is the most important when calculating scalar products between $\varphi_{wt}$ and $\theta_{td}$ in E-step formula (4). If considering the $\theta_{td}$ vector is dense (this is done to speed up the processing of the document and is not a significant problem since the vectors $\theta_d$ are stored only for documents from the current batch), the presence of indices of non-zero elements in $\varphi_{wt}$ allows proceeding only multiplications of elements with these indices. This preserves the locality of the operation in memory and avoids the use of heavyweight conditional operators.

To complete optimization procedure, we need to set a threshold for the ratio of non-zero elements. The rows, in which the ratio of non-zero values is above threshold, are stored in the dense form, otherwise in the sparse one. This threshold can be approximately estimated based on the requirement to prevent memory consumption in a sparse form greater than in a dense one. To do this, we formulate and solve the inequality: $2k + (m/8) < m \quad \Rightarrow \quad k < 0.4375\ m$. Based on this inequality, the value 0.4 was chosen as the maximum fraction of non-zero elements for storing in a sparse form.

## VI. Quality measures

Two quality measures are most often used for topic model evaluation. *Perplexity* is an inverse of the likelihood of data, i.e., the smaller it is, the better:

$$\mathcal{P}(\Phi, \Theta) = \exp\left( -\frac{1}{n} \sum_{d \in D} \sum_{w \in d} n_{dw} \ln p(w \mid d) \right).$$

The *coherence of a topic* $t$ is the average positive pointwise mutual information (PPMI) over term pairs:

$$\mathcal{C}_t(\Phi) = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \mathrm{PPMI}(w_i, w_j),$$

where $w_i$ is the $i$-th term in the list of $k$ most probable terms in the topic $t$, $\mathrm{PPMI}(u,v) = \max\{0, \ln \frac{|D|N_{uv}}{N_u N_v}\}$, $N_{uv}$ is the number of documents that contain terms $u$ and $v$ nearby, $N_u$ is the number of documents that contain the term $u$. The coherence is known to be highly correlated with expert judgments about topic interpretability [21]. Therefore the average coherence over topics is commonly used as an interpretability measure of the model calculated automatically without asking human experts.

## VII. Experiments

For the experiments, we use a workstation with Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz and 128Gb memory. All calculations are performed in eight threads. In the offline algorithm, we always use the collection of 200K documents sampled from English Wikipedia (http://docs.bigartm.org/en/stable/tutorials/datasets.html) in the Bag-of-Words format (100 batches), in the online ones — 1M documents from the same source (500 batches). Both collections share a dictionary of 100K tokens. In all experiments, we compare models with 100, 500, 1000 and 2000 topics.

Firstly we will show the positive effect of optimization, introduced in V, in terms of memory consumption and training time. Then using the optimized library we compare different normalization strategies that will be further proposed.

TABLE I. Optimization, offline algorithm. Columns: Topics — number of topics in model, Case — type of the model and training, Memory — peak memory consumption in megabytes, T. (X) — training time in seconds when train with X document passes. Best values are highlighted in bold font.

| Topics | Case | Memory | T. (1) | T. (5) | T. (10) | T. (15) |
|---|---|---|---|---|---|---|
| 100 | N/N | **630** | **110** | **165** | 230 | 295 |
| | N/Y | 710 | 120 | 170 | 240 | 305 |
| | Y/Y | 720 | 130 | 170 | **220** | **275** |
| 500 | N/N | **1010** | **350** | 605 | 940 | 1275 |
| | N/Y | 1220 | 370 | 635 | 965 | 1305 |
| | Y/Y | 1210 | 390 | **570** | **775** | **990** |
| 1000 | N/N | **1470** | **690** | 1265 | 1925 | 2595 |
| | N/Y | 1840 | 745 | 1305 | 1955 | 2640 |
| | Y/Y | 1800 | 755 | **1070** | **1410** | **1750** |
| 2000 | N/N | **2620** | **1735** | 2885 | 4140 | 5430 |
| | N/Y | 3190 | 1815 | 2945 | 4310 | 5530 |
| | Y/Y | 3160 | 1860 | **2585** | **3315** | **3960** |

### A. *Experiments with models sparsification*

In this set of experiments, we compare models

- without optimization for sparse models and without sparsification regularization (N/N);
- without optimization for sparse models and with sparsification regularization (N/Y);
- with optimization for sparse models and with sparsification regularization (Y/Y).

The training time and the consumption of memory are compared. To sparsify models, we use SmoothSparsePhi regularizer from BigARTM library with a default (uniform) strategy. We sparsify models to get as a result a 95-99% (depending on a number of topics) of zero elements in the final $\Phi$ matrix.

*1) Offline algorithm:* For the offline algorithm, we set the number of collection passes equal to 10. The number of document passes is selected from set 1, 5, 10, 15. The regularization coefficients for the models with 100, 500, 1000 and 2000 topics are $-1.0$, $-0.5$, $-0.5$, $-0.2$ respectively. We chose such values of the coefficients empirically as they allow to achieve a high sparsity of the $\Phi$ and $(n_{wt})$ matrices without significant loss of perplexity. The results are shown in table I. It is worth noting that hereinafter changes in the number of the document passes lead to minor (within 5%) fluctuations in peak memory consumption, so mean values are reported.

Sparse optimization does not give any improvement in model with 100 topics or with a training algorithm that uses only one document pass per collection pass. The model without the sparsification regularization (due to the absence of the $(r_{wt})$ matrix) has the lowest memory consumption. At the same time, the addition of sparsification combined with the optimization allows in different cases to obtain up to 30% acceleration compared to the dense model.

*2) Online algorithm:* In experiments with the online algorithm, we use only one collection pass, while the number of document passes is selected from set 1, 5, 10, 15 again. Merging $(n_{wt})$ matrices with following normalization and regularization starts up every 32 processed batches. The regularization coefficients for the models with 100, 500, 1000, and 2000 topics, in this case, are $-0.1$, $-0.03$, $-0.015$, $-0.01$, respectively.

TABLE II. Optimization, online algorithm. Columns: Topics — number of topics in model, Case — type of the model and training, Memory — peak memory consumption in megabytes, T. (X) — training time in seconds when train with X document passes. Best values are highlighted in bold font.

| Topics | Case | Memory | T. (1) | T. (5) | T. (10) | T. (15) |
|---|---|---|---|---|---|---|
| 100 | N/N | **900** | **85** | 115 | 185 | 230 |
| | N/Y | 990 | 95 | 125 | 165 | 200 |
| | Y/Y | 950 | 90 | **110** | **150** | **160** |
| 500 | N/N | **1880** | 410 | 515 | 770 | 1010 |
| | N/Y | 2100 | 395 | 495 | 805 | 1015 |
| | Y/Y | 1950 | **390** | **450** | **650** | **700** |
| 1000 | N/N | **3150** | 720 | 900 | 1200 | 1785 |
| | N/Y | 3600 | 740 | 940 | 1270 | 1780 |
| | Y/Y | 3270 | **715** | **820** | **950** | **1245** |
| 2000 | N/N | 5800 | 1330 | 1820 | 2580 | 3650 |
| | N/Y | 6580 | 1380 | 1890 | 2630 | 3315 |
| | Y/Y | **5590** | **1200** | **1510** | **1840** | **2210** |

TABLE III. Optimization, asynchronous online algorithm. Columns: Topics — number of topics in model, Case — type of the model and training, Memory — peak memory consumption in megabytes, T. (X) — training time in seconds when train with X document passes. Best values are highlighted in bold font.

| Topics | Case | Memory | T. (1) | T. (5) | T. (10) | T. (15) |
|---|---|---|---|---|---|---|
| 100 | N/N | 1170 | 65 | 80 | 120 | 150 |
| | N/Y | 1250 | 65 | 85 | 125 | 150 |
| | Y/Y | **1110** | **60** | **75** | **110** | **125** |
| 500 | N/N | 2700 | 200 | 340 | 515 | 660 |
| | N/Y | 2980 | 190 | 340 | 535 | 670 |
| | Y/Y | **2270** | **185** | **290** | **450** | **500** |
| 1000 | N/N | 4640 | 520 | 730 | 1095 | 1435 |
| | N/Y | 5230 | 555 | 760 | 1115 | 1475 |
| | Y/Y | **3910** | **500** | **600** | **900** | **1065** |
| 2000 | N/N | 9110 | 860 | 1400 | 2140 | 2730 |
| | N/Y | 9760 | 865 | 1415 | 2120 | 2720 |
| | Y/Y | **7230** | **780** | **1120** | **1670** | **2010** |

Table II shows the result of comparison. As for the offline algorithm, optimization does not give any advantage in the case of models with a small number of topics and algorithms with one document pass. In other cases, it also gives up to 30% reduction in computation time compared to a dense model. An additional advantage is that, unlike the offline algorithm, in the case of online, sparse storage-processing can save memory on the storage of intermediate versions of the $(n_{wt})$ matrix. Due to this, usage of regularization does not lead to increased consumption of memory compared to N/N algorithm.

*3) Asynchronous online algorithm:* All parameters of the experiments with asynchronous online algorithm use the same set of the parameters as synchronous one. The results are shown in table III. In all cases the asynchronous algorithm obtains up to 25% acceleration compared to the dense model. Sparse storage-processing reduces memory consumption in the case of a regularized model (saving up to 23% in various cases) compared to a dense model without regularization.

### *B. Experiments with E-step normalization*

In the main set of experiments, we compare different strategies of fast E-step usage. The combination of fast and ordinary steps can be done along the collection passes (i.e., part of the passes should be done completely with the usual steps, part — with fast), but it is also possible to use it during document passes (both types of E-steps can be used while one document processing). We call the described sets of strategies, external and internal, respectively. As part of the experiment, it is proposed in each group to compare five types of strategies:

TABLE IV. Normalization strategies, offline algorithm. Columns: Topics — number of topics in model, Case — model type/E-steps combination strategy, Perplexity — final perplexity value, Coherence — final coherence value, OMT (on min time) — metric value on the moment of the fastest case finish, Time — time to finish all collection passes in seconds. Best results are highlighted in bold font.

| Topics | Case | Perplexity [OMT] | Coherence [OMT] | Time |
|---|---|---|---|---|
| 100 | N/F | 3729 [3769] | 0.086 [0.083] | 720 |
| | N/N | 5333 [5333] | 0.104 [0.104] | 510 |
| | N/M | 4038 [4357] | 0.088 [0.098] | 630 |
| | N/H | 3747 [3774] | 0.084 [0.083] | 620 |
| | N/L | 3827 [3874] | 0.087 [0.083] | 555 |
| | **D/F** | **3741 [3795]** | **0.092 [0.090]** | **760** |
| 500 | N/F | 2076 [2123] | 0.092 [0.093] | 2850 |
| | N/N | 3759 [3759] | 0.098 [0.098] | 1740 |
| | N/M | 2442 [2811] | 0.103 [0.103] | 2375 |
| | **N/H** | **2077 [2121]** | **0.093 [0.097]** | **2300** |
| | N/L | 2169 [2264] | 0.095 [0.096] | 1945 |
| | D/F | 2101 [2169] | 0.094 [0.091] | 3570 |
| 1000 | N/F | 1483 [1546] | 0.078 [0.078] | 5430 |
| | N/N | 3122 [3122] | 0.088 [0.088] | 3295 |
| | N/M | 1845 [2198] | 0.092 [0.094] | 4465 |
| | **N/H** | **1474 [1498]** | **0.084 [0.084]** | **4224** |
| | N/L | 1561 [1612] | 0.084 [0.085] | 3672 |
| | D/F | 1571 [1619] | 0.077 [0.077] | 6649 |
| 2000 | N/F | 998 [1025] | 0.057 [0.059] | 11405 |
| | N/N | 2574 [2574] | 0.069 [0.069] | 7775 |
| | N/M | 1298 [1618] | 0.071 [0.074] | 9975 |
| | **N/H** | **977 [1006]** | **0.063 [0.063]** | **9380** |
| | N/L | 1049 [1129] | 0.064 [0.065] | 8570 |
| | D/F | 1255 [1524] | 0.056 [0.057] | 28500 |

- FULL (F): all iterations are normal;
- NONE (N): all iterations are fast;
- MIXED (M): fast and normal iterations alternate, the last iteration is always normal;
- HALF (H): the first half of the iterations is fast, the second is the usual one;
- LAST (L): 80% of the first iterations are fast, the rest are the usual ones.

Models are evaluated based on a set of criteria: training speed, perplexity, and coherence. In order to evaluate coherence, the co-occurrence counters were collected over 1M documents of English Wikipedia. The counter for the token pair was incremented by one for each document where these tokens appear at least once. Counters with value less than 100 were resets to 0. In addition to each other, all the described methods are also compared to usual (i.e. FULL) models with uniform sparsification (SPARSE, S) or topics decorrelation (DECOR, D), provided by SmoothSparsePhi and DecorrelatorPhi regularizers from BigARTM library respectively. Let us denote the model without regularization as NONE (N).

*1) Offline algorithm:* For the offline algorithm, we conducted a preliminary experiment in which we estimated the value of perplexity for the same time spent with a different number of document passes. As a result, it turned out that 5 document passes are optimal from this point of view. We use this value in further experiments. The number of collection passes is set to 40 (with this number of passes, by the end of processing, perplexity almost stop changing). For an offline algorithm, we study only external sets of strategies for combining steps; internal one's usage does not make much sense as we can process the same document many times during different collection passes.

We tried several different regularization coefficients for topic decorrelation regularization. The best results were obtained using the coefficients 1e+6, 2e+5, 1e+5, 5e+4 for 100, 500, 1000, and 2000, respectively.

In the primary set of the experiments we measure two types of perplexity and coherence values: one pair of numbers is calculated at the time of completion of 40 collection passes, the second one is obtained when a fixed allocated time has passed. For each number of topics, the value of time interval is determined by the training time of the fastest algorithm.

The table IV shows that

- the decorrelation without fast E-steps is the best choice in case of model with 100 topics;
- the decorrelation is failing in all other cases because computation time of this regularizer increases quadratically while number of topics grows;
- the NONE strategy is obviously the fastest one, and also improves coherence, but it spoils perplexity very much;
- the HALF strategy is an optimal choice both in case of fixed number of collection passes (the train time saving is up to 20%) and in case of processing until the end of the time interval; anyway we save or even decrease final perplexity value and achieve up to 10% coherence growth.

It is worth saying that experiments with uniform sparsification were not included in the results table, as well as some ones with the parallel usage of regularizers and mixed E-step strategies, because they didn't yield any positive results.

*2) Online algorithm:* Online algorithms are designed for on-the-fly collections processing, so the number of collection passes is set to 1. In this case, we can increase the number of document passes to achieve better perplexity, but it is expected to increase training time. As a compromise, the number of document passes is fixed and is equal to 10 for all experiments. The frequency of the model updates remains the same as in previous experiments with online algorithms (once per every 32 processed batches).

As in the case of the offline algorithm, we want not only to obtain acceleration when using mixed normalization strategies at the E-step, but also be able to use this gain in order to try to improve the quality of the model due to it. Since we cannot avoid processing of some part of the collection, it is proposed to use the time gained for training with more frequent model updates (every 24, 16 and 8 batches).

For the online algorithm, as well as for the offline one, the coefficients of the decorrelation regularizer were cross-validated. The set 1e+5, 5e+3, 2e+3, and 1e+3 for 100, 500, 1000, and 2000 topics, respectively, turned out to be the best one. In the regularizer of uniform sparsification, the coefficients 0.1, 0.03, 0.015, 0.01 are used for the same set of numbers of topics.

TABLE V. NORMALIZATION STRATEGIES, ONLINE ALGORITHM. COLUMNS: TOPICS — NUMBER OF TOPICS IN MODEL, CASE — MODEL TYPE/E-STEPS COMBINATION STRATEGY, PERPLEXITY — FINAL PERPLEXITY VALUE, COHERENCE — FINAL COHERENCE VALUE, OMT (ON MIN TIME) — METRIC VALUE ON THE MOMENT OF THE FASTEST CASE FINISH, TIME — TIME TO FINISH ALL COLLECTION PASSES IN SECONDS. BEST RESULTS ARE HIGHLIGHTED IN BOLD FONT.

| Topics | Case | Update every | Perplexity | Coherence | Time |
|---|---|---|---|---|---|
| 100 | N/F | 32 | 8983 | 0.091 | 180 |
| | N/N | 32 | 10323 | 0.074 | 100 |
| | N/M | 32 | 9138 | 0.084 | 135 |
| | N/H | 32 | 9033 | 0.081 | 135 |
| | N/L | 32 | 9721 | 0.077 | 115 |
| | S/F | 32 | 8891 | 0.066 | 155 |
| | **D/F** | **32** | **8862** | **0.128** | **190** |
| | N/H | 24 | 8104 | 0.077 | 155 |
| | N/H | 16 | 7031 | 0.084 | 190 |
| | N/H | 8 | 6030 | 0.095 | 285 |
| 500 | N/F | 32 | 6630 | 0.080 | 710 |
| | N/N | 32 | 9427 | 0.047 | 365 |
| | N/M | 32 | 7342 | 0.071 | 540 |
| | N/H | 32 | 7030 | 0.076 | 545 |
| | N/L | 32 | 8314 | 0.056 | 440 |
| | S/F | 32 | 6563 | 0.056 | 525 |
| | D/F | 32 | 6401 | 0.077 | 820 |
| | N/H | 24 | 6189 | 0.075 | 620 |
| | **N/H** | **16** | **5265** | **0.080** | **750** |
| | N/H | 8 | 4442 | 0.071 | 1120 |
| 1000 | N/F | 32 | 5871 | 0.063 | 1435 |
| | N/N | 32 | 9113 | 0.044 | 835 |
| | N/M | 32 | 6712 | 0.055 | 1145 |
| | N/H | 32 | 6315 | 0.057 | 1120 |
| | N/L | 32 | 7789 | 0.050 | 955 |
| | S/F | 32 | 5663 | 0.051 | 1065 |
| | D/F | 32 | 5558 | 0.053 | 2110 |
| | N/H | 24 | 5519 | 0.057 | 1215 |
| | **N/H** | **16** | **4652** | **0.060** | **1475** |
| | N/H | 8 | 3890 | 0.058 | 2175 |
| 2000 | N/F | 32 | 5144 | 0.045 | 2995 |
| | N/N | 32 | 8879 | 0.038 | 1630 |
| | N/M | 32 | 6145 | 0.046 | 2315 |
| | N/H | 32 | 5694 | 0.046 | 2320 |
| | N/L | 32 | 7324 | 0.042 | 1911 |
| | S/F | 32 | 4920 | 0.040 | 2145 |
| | D/F | 32 | 7211 | 0.031 | 12780 |
| | N/H | 24 | 4962 | 0.045 | 2595 |
| | **N/H** | **16** | **4157** | **0.044** | **3155** |
| | N/H | 8 | 3428 | 0.039 | 4589 |

Note that in this paper, we present the results of the experiments with both versions of the online algorithm only for the group of internal strategies for E-steps mixing. We do it since the external ones (in which some documents are processed with normalization at the E-step and some — without it) failed significantly compared to all cases.

The results of the experiments for an online synchronous algorithm and group of internal E-step mixing strategies are shown in table V. According to it, we can do the following conclusions:

- the decorrelation is the best solution for a model with 100 topics and ill-suited one for large models again;
- usage of mixing strategies with the same frequency of model updates reduces train time by 25-50%, but worsen both perplexity and coherence significantly;
- usage of the most perspective strategy HALF with more frequent updates allows us to gain a big (up to 23%) improvement of the perplexity with small decay of speed and coherence or even without it.

Our attempts to use the time gained while usage of uniform sparsification due to optimization, described in V, failed for the synchronous online algorithm, as more frequent updates decreased the already spoiled coherence significantly.

TABLE VI. NORMALIZATION STRATEGIES, ASYNCHRONOUS ONLINE ALGORITHM. COLUMNS: TOPICS — NUMBER OF TOPICS IN MODEL, CASE — MODEL TYPE/E-STEPS COMBINATION STRATEGY, PERPLEXITY — FINAL PERPLEXITY VALUE, COHERENCE — FINAL COHERENCE VALUE, OMT (ON MIN TIME) — METRIC VALUE ON THE MOMENT OF THE FASTEST CASE FINISH, TIME — TIME TO FINISH ALL COLLECTION PASSES IN SECONDS. BEST RESULTS ARE HIGHLIGHTED IN BOLD FONT.

| Topics | Case | Update every | Perplexity | Coherence | Time |
|---|---|---|---|---|---|
| 100 | N/F | 32 | 13391 | 0.070 | 115 |
| | N/N | 32 | 13856 | 0.054 | 60 |
| | N/M | 32 | 13262 | 0.066 | 90 |
| | N/H | 32 | 13308 | 0.064 | 90 |
| | N/L | 32 | 13545 | 0.061 | 70 |
| | S/F | 32 | 12258 | 0.065 | 100 |
| | D/F | 32 | 13619 | 0.096 | 115 |
| | N/H | 24 | 11374 | 0.067 | 90 |
| | N/H | 16 | 9373 | 0.077 | 95 |
| | N/H | 8 | 7197 | 0.083 | 125 |
| | S/F | 24 | 10638 | 0.063 | 100 |
| | S/F | 16 | 9112 | 0.057 | 100 |
| | S/F | 8 | 7554 | 0.039 | 130 |
| | D/F | 24 | 11630 | 0.100 | 115 |
| | **D/F** | **16** | **9614** | **0.134** | **120** |
| | D/F | 8 | 7586 | 0.105 | 170 |
| 500 | N/F | 32 | 11034 | 0.070 | 490 |
| | N/N | 32 | 12716 | 0.041 | 185 |
| | N/M | 32 | 11385 | 0.059 | 340 |
| | N/H | 32 | 11273 | 0.060 | 340 |
| | N/L | 32 | 12024 | 0.047 | 245 |
| | S/F | 32 | 9762 | 0.057 | 365 |
| | **D/F** | **32** | **11007** | **0.093** | **495** |
| | N/H | 24 | 9265 | 0.071 | 340 |
| | N/H | 16 | 7343 | 0.075 | 355 |
| | **N/H** | **8** | **5400** | **0.076** | **455** |
| | S/F | 24 | 8166 | 0.054 | 345 |
| | S/F | 16 | 6683 | 0.050 | 345 |
| | S/F | 8 | 5377 | 0.036 | 415 |
| | **D/F** | **24** | **8892** | **0.079** | **490** |
| | **D/F** | **16** | **6908** | **0.079** | **540** |
| | D/F | 8 | 5137 | 0.063 | 810 |
| 1000 | N/F | 32 | 10154 | 0.062 | 1010 |
| | N/N | 32 | 12344 | 0.039 | 380 |
| | N/M | 32 | 10652 | 0.054 | 710 |
| | N/H | 32 | 10456 | 0.054 | 710 |
| | N/L | 32 | 11452 | 0.042 | 520 |
| | S/F | 32 | 8796 | 0.054 | 750 |
| | D/F | 32 | 9895 | 0.061 | 1040 |
| | N/H | 24 | 8472 | 0.057 | 700 |
| | **N/H** | **16** | **6599** | **0.056** | **710** |
| | **N/H** | **8** | **4788** | **0.058** | **900** |
| | S/F | 24 | 7161 | 0.054 | 710 |
| | S/F | 16 | 5716 | 0.047 | 680 |
| | S/F | 8 | 4421 | 0.032 | 785 |
| | D/F | 24 | 7856 | 0.057 | 1175 |
| | D/F | 16 | 5935 | 0.052 | 1675 |
| | D/F | 8 | 4346 | 0.044 | 3280 |
| 2000 | N/F | 32 | 9267 | 0.049 | 2150 |
| | N/N | 32 | 12024 | 0.033 | 1050 |
| | N/M | 32 | 9969 | 0.045 | 1590 |
| | N/H | 32 | 9707 | 0.045 | 1540 |
| | N/L | 32 | 10902 | 0.037 | 1225 |
| | **S/F** | **32** | **7821** | **0.047** | **1560** |
| | D/F | 32 | 8654 | 0.040 | 5435 |
| | **N/H** | **24** | **7748** | **0.046** | **1590** |
| | **N/H** | **16** | **5976** | **0.044** | **1615** |
| | N/H | 8 | 4274 | 0.043 | 1890 |
| | **S/F** | **24** | **6236** | **0.045** | **1470** |
| | S/F | 16 | 4855 | 0.038 | 1390 |
| | S/F | 8 | 3921 | 0.025 | 1550 |

*3) Asynchronous online algorithm:* Parameters of the experiments for the asynchronous online algorithm are identical to ones for synchronous algorithm. To understand the results, remember that the asynchronous algorithm performs normalization and regularization operations in the background, in parallel with the processing of documents. This enhances the effect of any acceleration on the E-step and allows algorithm to ignore the costs of regularization until the time taken to normalize and regularize does not exceed the processing time of a set of batches between two model updates.

The results of experiments with E-steps mixing strategies for asynchronous algorithm given in the table VI show that

- decorrelation is the most suitable choice for models with 100 and 500 topics and is useless due to its slow speed and low model quality in case of 1000 and 2000 topics;
- fast E-step usage gives even greater acceleration compared to synchronous algorithm (up to two times for NONE strategy), but the model quality with the same updates frequency is worse than the in case of basic algorithm;
- HALF strategy with more frequent model updates is a best strategy for models with 1000 and 2000 topics: with maximum coherence losses within 10%, it allows to get more than twice the gain in perplexity, in all cases remaining faster than the basic algorithm by 10–30%;
- the uniform sparsification for a number of cases allows to obtain a lower perplexity value for all update frequencies compared to the HALF strategy without regularization, and it works even faster due to optimization for sparse models, but more frequent model updates, necessary to improve perplexity, in this case spoil the coherence very much (up to 80%).

## VIII. KEY FINDINGS AND RECOMMENDATIONS

The list of conclusions, described in the previous section, is quite extensive, so here we would like to provide a short set of recommendations on the use of the proposed improvements. It is worth noting that they fit models that are trained on relatively large text collections (like the one we use in this paper).

The hybrid storage format is useful in all cases. Optimized computation for sparse models is the best choice in case of models with more than 100 topics, that are trained with uniform sparsification and several document passes.

Fast E-steps are almost useless in case of small (less than 500 topics) models, as it is possible to achieve better results using carefully tuned topics decorrelation regularizer. When training model with a greater number of topics using the offline algorithm the external HALF strategy with a reduced number of collection passes is an optimal solution. In the case of online algorithms, the most suitable choice is an internal HALF strategy with 1.5-2 times higher $\Phi$ matrix updates rate.

## IX. CONCLUSION

The contribution of this paper is twofold. First, we generalize the EM-algorithm for any differentiable loss function in an optimized functional when training topic models. Second, we find experimentally the superior strategy for combining normal and fast E-steps, that allows us to achieve better train speed and higher model quality in most cases. This strategy is used for a different types of passes and provides significantly different results when training with offline and online EM-algorithm.

We consider BigARTM optimization for sparse models to be an additional result of our work. Also, we have discovered some new properties of the topic decorrelation regularizer, that extend previous ones [14] to the cases of online algorithm and models with many topics.

The future work includes more intensive study of regularizers and mixing E-step strategies combinations, as well as studying loss functions, other than linear and logarithmic ones.

## Acknowledgment

The work was supported by Russian Ministry of education and science (Project 5-100, contract 05.Y09.21.0018) and the Russian Foundation for Basic Research (grant 20-07-00936).

## References

[1] D. M. Blei, "Probabilistic topic models," *Communications of the ACM*, vol. 55, no. 4, pp. 77–84, 2012.

[2] J. Boyd-Graber, Y. Hu, and D. Mimno, *Applications of Topic Models*, ser. Foundations and Trends(r) in Information Retrieval Series. now Publishers Incorporated, 2017.

[3] T. Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 1999, pp. 50–57.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[5] H. Jelodar, Y. Wang, C. Yuan, X. Feng, X. Jiang, Y. Li, and L. Zhao, "Latent dirichlet allocation (lda) and topic modeling: models, applications, a survey," *Multimedia Tools and Applications*, vol. 78, no. 11, pp. 15 169–15 211, 2019.

[6] A. Ianina and K. Vorontsov, "Regularized multimodal hierarchical topic model for document-by-document exploratory search," in *Proceeding Of The 25th Conference Of FRUCT (Finnish-Russian University Cooperation in Telecommunications) Association. The seminar on Intelligence, Social Media and Web (ISMW). Helsinki, Finland, November 5–8, 2019.*, S. Balandin, V. Niemi, and T. Tutina, Eds., 2019, pp. 131–138.

[7] S. I. Nikolenko, S. Koltcov, and O. Koltsova, "Topic modelling for qualitative studies," *Journal of Information Science*, vol. 43, no. 1, pp. 88–102, 2017.

[8] R. Wesslen, "Computer-assisted text analysis for social science: Topic models and beyond," *CoRR*, vol. abs/1803.11045, 2018.

[9] M. J. Paul and M. Dredze, "Discovering health topics in social media using topic models," *PLoS ONE*, vol. 9, no. 8, 2014.

[10] J. O'Neill, C. Robin, L. O'Brien, and P. Buitelaar, "An analysis of topic modelling for legislative texts," in *Proceedings of the Second Workshop on Automated Semantic Analysis of Information in Legal Texts co-located with the 16th International Conference on Artificial Intelligence and Law (ICAIL 2017)*, 2017.

[11] M. Reisenbichler and T. Reutterer, "Topic modeling in marketing: recent advances and research opportunities," *Journal of Business Economics*, vol. 89, no. 3, pp. 327–356, 2019.

[12] O. Khalifa, D. Corne, M. Chantler, and F. Halley, "Multi-objective topic modelling," in *7th International Conference Evolutionary Multi-Criterion Optimization (EMO 2013)*. Springer LNCS, 2013, pp. 51–65.

[13] K. V. Vorontsov, "Additive regularization for topic models of text collections," *Doklady Mathematics*, vol. 89, no. 3, pp. 301–304, 2014.

[14] K. V. Vorontsov and A. A. Potapenko, "Additive regularization of topic models," *Machine Learning, Special Issue on Data Analysis and Intelligent Optimization with Applications*, vol. 101, no. 1, pp. 303–323, 2015.

[15] D. Kochedykov, M. Apishev, L. Golitsyn, and K. Vorontsov, "Fast and modular regularized topic modelling," in *Proceeding Of The 21St Conference Of FRUCT (Finnish-Russian University Cooperation in Telecommunications) Association. The seminar on Intelligence, Social Media and Web (ISMW). Helsinki, Finland, November 6-10, 2017*. IEEE, 2017, pp. 182–193.

[16] K. Vorontsov, O. Frei, M. Apishev, P. Romov, M. Suvorova, and A. Yanina, "Non-bayesian additive regularization for multimodal topic modeling of large collections," in *Proceedings of the 2015 Workshop on Topic Models: Post-Processing and Applications*. New York, NY, USA: ACM, 2015, pp. 29–37.

[17] Y. Tan and Z. Ou, "Topic-weak-correlated latent Dirichlet allocation," in *7th International Symposium Chinese Spoken Language Processing (ISCSLP)*, 2010, pp. 224–228.

[18] O. Frei and M. Apishev, "Parallel non-blocking deterministic algorithm for online topic modeling," in *AIST'2016, Analysis of Images, Social networks and Texts*, vol. 661. Springer International Publishing Switzerland, Communications in Computer and Information Science (CCIS), 2016, pp. 132–144.

[19] J. Yuan, F. Gao, Q. Ho, W. Dai, J. Wei, X. Zheng, E. P. Xing, T.-Y. Liu, and W.-Y. Ma, "LightLDA: Big topic models on modest computer clusters," in *WWW*, 2015, pp. 1351–1361.

[20] B. Zhao, H. Zhou, G. Li, and Y. Huang, "ZenLDA: An efficient and scalable topic model training system on distributed data-parallel platform," *CoRR*, vol. abs/1511.00440, 2015.

[21] D. Newman, Y. Noh, E. Talley, S. Karimi, and T. Baldwin, "Evaluating topic models for digital libraries," in *Proceedings of the 10th annual Joint Conference on Digital libraries*, ser. JCDL '10. New York, NY, USA: ACM, 2010, pp. 215–224.

## Appendix

The proof of the Theorem 2.

The Karush–Kuhn–Tucker (KKT) necessary conditions for the local extremum $(\Phi, \Theta)$ of the problem (7), (3) can be written as follows:

$$\sum_{d\in D} n_{dw}\theta_{td}\ell'(p(w\,|\,d)) + \frac{\partial R}{\partial \varphi_{wt}} - \lambda_t + \lambda_{wt} = 0; \tag{9}$$

$$\lambda_{wt} \geqslant 0; \quad \lambda_{wt}\varphi_{wt} = 0;$$

$$\sum_{w\in W} n_{dw}\varphi_{wt}\ell'(p(w\,|\,d)) + \frac{\partial R}{\partial \theta_{td}} - \mu_d + \mu_{td} = 0; \tag{10}$$

$$\mu_{td} \geqslant 0; \quad \mu_{td}\theta_{td} = 0;$$

where $\lambda_t$ and $\mu_d$ are KKT multipliers for normalization constraints, $\lambda_{wt}$ and $\mu_{td}$ are KKT multipliers for nonnegativity constraints.

Let us multiply both sides of equation (9) by $\varphi_{wt}$ and substitute the auxiliary variable $p_{tdw}$ from (8). Then we substitute the sum over $d \in D$ by $n_{wt}$ auxiliary variable from (5). The calculations for $\theta_{td}$ variables from (10) are analogous:

$$\varphi_{wt}\lambda_t = \sum_{d\in D} n_{dw}p_{tdw} + \varphi_{wt}\frac{\partial R}{\partial \varphi_{wt}} = n_{wt} + \varphi_{wt}\frac{\partial R}{\partial \varphi_{wt}};$$

$$\theta_{td}\mu_d = \sum_{w\in W} n_{dw}p_{tdw} + \theta_{td}\frac{\partial R}{\partial \theta_{td}} = n_{td} + \theta_{td}\frac{\partial R}{\partial \theta_{td}}.$$

An assumption that $\lambda_t \leqslant 0$ ($\mu_d \leqslant 0$) leads to a degenerate case when the $t$-th column of the $\Phi$ matrix ($d$-th column of the $\Theta$ matrix) equals zero. Therefore, our main case is $\lambda_t > 0$ ($\mu_d > 0$). Using the nonnegativity condition $\varphi_{wt} \geqslant 0$ ($\theta_{td} \geqslant 0$) we write:

$$\varphi_{wt}\lambda_t = \max\Bigl\{n_{wt} + \varphi_{wt}\frac{\partial R}{\partial \varphi_{wt}}, 0\Bigr\}; \tag{11}$$

$$\theta_{td}\mu_d = \max\Bigl\{n_{td} + \theta_{td}\frac{\partial R}{\partial \theta_{td}}, 0\Bigr\}. \tag{12}$$

Let us sum both sides of the first equation over $w \in W$, then both sides of the second equation over $t \in T$:

$$\lambda_t = \sum_{w\in W^m} \max\Bigl\{n_{wt} + \varphi_{wt}\frac{\partial R}{\partial \varphi_{wt}}, 0\Bigr\}; \tag{13}$$

$$\mu_d = \sum_{t\in T} \max\Bigl\{n_{td} + \theta_{td}\frac{\partial R}{\partial \theta_{td}}, 0\Bigr\}. \tag{14}$$

We obtain (5) by expressing $\varphi_{wt}$ from (11) and (13); then we obtain (6) by expressing $\theta_{td}$ from (12) and (14). This completes the proof of the theorem.