# Topic Models Selection for Reading Orders Generation over Document Collections

***Mamonov K. R., Vorontsov K. V., Eremeev M. A***
mamonov.kr@phystech.ru, vokov@forecsys.ru, maks5507@yandex.ru
Moscow Institute of Physics and Technology

This paper investigates an effect of Topic Models Selection on the algorithm that automatically organizes a collection of documents in a tree from general to more specific documents, and allows a user to choose a reading sequence over the documents. We evaluate the impact of ARTM technique and its variations, hierarchical ARTM and multimodal ARTM, on the algorithm. Such alterations enhance the overall quality as well as improving the efficiency of the algorithm. We study new approaches in measuring document generality, featuring new metric named hierarchical entropy. The experiments were conducted with Russian Wikipedia hierarchies of categories in Mathematics and Machine Learning. The problem to solve gives a novel way to content consumption that is significant for learning or editorial purposes.

**Keywords**: *reading trees generation, reading sequence, probabilistic topic modeling* .

## 1 Introduction

While simple search engines like Google or Yahoo rank search results based on the relevance to the provided query. That requires the user to know the area and be able to formulate the query using specific terms. As for users wanting to explore the new topic, this kind of search and ranking approaches do not work effectively.

Exploratory search ( [30], [20]) is a paradigm of information retrieval, in which the user intends to learn the subject domain better. As the user does not know the main concepts of the domain, he needs a hint which of the found documents to read first, gradually moving from simple to more complex documents.

Reading order optimization is an alternative way to content consumption that departs from the typical ranked lists of documents based on their relevance. Given a document collection, existing systems allow users to browse the collection or perform searches that return lists of documents ranked based on their relevance to the user query. While these approaches work fine when a user is trying to locate specific documents, they are insufficient when users need to access the pertinent documents in some logical order, for example, for learning or editorial purposes. Such a ranking technique helps to achieve more accurate processing of information significantly faster.

Actually, this problem is not widely covered in literature. Attempts to convert document collection into some meaningful structure have already been in [23] and [11], but this approaches applicable only for specified document collections, whereas our aim is to develop a method for arbitrary collections; nevertheless, [17] explores the possible solution to this problem, which we elaborate in this paper. The proposed algorithm is based on the more flexible clustering techniques than used previously in label tree learning methods( [3], [18], [19]) and vector representations of documents, obtained by topic modeling [13]. Authors suppose the reading order is not a list, but a set of trees. Indeed, when exploring the topic, users sometimes have to make a decision of what subarea to dive into, as the amount of information is vast.

Applications, such as automatic curriculum generation, are discussed in [16] and [22]; however, they do not provide any scalable solutions, only tailored variations.

Multiple approaches on counting models generality and complexity were proposed in [10] and [24]. In order to solve the problem, they count simple characteristics and sort the documents by them. For instance, the complexity of the document can be calculated effectively ( [10]), but sorting does not provide tree structure, plain lists only.

[14] explores methods of deriving base documents using variations of PageRank ( [8]) algorithm. The proposed methods are query-based, meaning that the user has to provide a query to get a sorted list of documents. Algorithms cannot be applied to sort arbitrary document collection, without a given query.

In our paper, we extend approach, proposed in [17], by trying to vary the topic models, which are the core of the algorithm. We scrutinize the ARTM ( [29], [27]) approach, which is proved to outperform LDA [5] in many tasks due to the wide choice of regularizers accessible [26]. We compare different types of models, including hierarchical models ( [7], [2]) with diverse combinations of regularizers, as well as with different text preprocessing methods like terms and bigrams extractions.

There is a lot of approaches to model the hierarchical nature of topics. For example, hLDA [4] presents a way of topical hierarchy as a tree with one parent for each sub-topic, hPAM [21] overcomes this limitation and represents hierarchies as multilevel acyclic graphs, hHDP [31] additionally provide ways to estimate the number of levels and number of topics on each level of hierarchy and use multilevel graph model too. In this work, we use hARTM because ща the better results shown in experiments [2].

Moreover, we analyze possible drawbacks of the algorithm, proposing extensions to overcome them, e.g., we introduce hierarchical entropy, successfully upgrading accuracy of generality estimation.

## 2 Problem statement

A reading graph $R(V, E)$ over a document set $D$ is a directed acyclic graph whose nodes correspond to the input documents and edges capture specificity relations among the documents. In particular, a node $v_i \in V$ maps a non-empty set $D_i \subseteq D$ of equivalent documents. An edge $v_i \to v_j$ between nodes $v_i$ and $v_j$ signifies that documents belonging to the corresponding document set $D_i$ precede documents belonging to the respective set $D_j$.

A complete reading tree over a document set $D$ is a reading graph $R(V, E)$ with the following properties:

(a) For each node $v_i \in V$ with $D_i$ being its corresponding set of documents, it holds that: a document $d \in D$ maps to $v_i \Leftrightarrow d \leftrightarrow d_i$ , for all $d_i \in Di$.

(b) For each pair of nodes $v_i, v_j \in V$ with $D_i$ and $D_j$ being their sets of documents, and an edge $v_i \to v_j$ , it holds that: For each pair of documents $d_i \in D_i$ and $d_j \in D_j$ , it is $d_i \to d_j$.

(c) For each pair of nodes $v_i, v_j \in V$ with $D_i$ and $D_j$ being their sets of documents, and an edge $v_i \to v_j$ , it holds that: there is no other node $v_k$, such that: $v_i \to v_k \to v_j$.

A reading sequence $d_{m1} \to d_{m2} \ldots \to d_{mk}$ of documents $d_{mi} \in D, i = 1 \ldots k$, maps to a path $v_{l1} \to v_{l2} \ldots \to v_{lk}$ on the graph with $v_{li} \in V, i = 1 \ldots k$ such that $d_{mi} \in D_{li}$ of node $v_{li}, i = 1 \ldots k$.

In order to compare and evaluate the reading graph, we need a way to compare trees. Edit distance metrics, initially introduced for string comparison, have been used to compare ordered trees [25]. Ordered labeled trees are trees in which the left-to-right order among siblings is significant. A distance between two trees is computed by considering an optimal mapping between two trees as the minimum cost of a sequence of elementary operations that converts one tree into the other. An alternative to mapping and tree edition is tree alignment [15].

Our reading order problem is different, and thus we are not interested in how identical two trees are. We care for the relative ordering of each pair of documents. To quantify the tree difference based on the pairwise document orderings, we first build the adjacency matrix A for a tree structure using the following formula:

$$A_{ij} = \begin{cases} \frac{1}{num_{hops}(d_i \to d_j)} & \text{if there is a directed path } d_i \to d_j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

To measure the difference of two tree structures over a set of documents represented by matrixes $A$ and $\hat{A}$, we use the mean squared error (MSE), which is defined as:

$$\frac{1}{n}\sum_{i,j=1}^{n}(A_{ij} - \hat{A_{ij}})^2 \tag{2}$$

## 3 Model Relationships Between Documents

### 3.1 Topic Modeling

As described above, the foundation of the algorithm is document embeddings, i.e., semantic vector representations [1]. Probabilistic topic modeling [13] is a technique used to build the probabilistic document embeddings with the interpretive components. Each component $i$ of the document $d$ representation is considered the probability of the document $d$ belonging to the topic $i$. Hence, the topic model simultaneously calculates document embeddings and performs soft clustering.

Topic modeling central assumption is that the probability of the word $w$ occurs in the document $d$:

$$p(w \mid d) = \sum_{t \in T} p(w \mid t)p(t \mid d) = \sum_{t \in T} \varphi_{wt}\theta_{td} \tag{3}$$

where matrix $\Phi$ contains distributions of word $w$ in document $d$ ($\varphi_{wt}$), and matrix $\Theta$ – probabilities $\theta_{td}$ of topic $t$ occurs in document $d$. $T$ is the total number of topics in the model.

The primary model proposed in [13] is **Probabilistic Latent Semantic Analysis (PLSA)** obtains the solutions matrices by maximizing the likelihood:

$$L(\Phi, \Theta) = \sum_{d \in D}\sum_{w \in d} n_{wd} \log p(w \mid d) \to \max_{\Phi,\Theta} \tag{4}$$

With the constraints on $\Phi$ and $\Theta$:

$$\sum_{w \in W} \varphi_{wt} = 1, \varphi_{wt} \geqslant 0 \tag{5}$$

$$\sum_{t \in T} \theta_{td} = 1, \theta_{td} \geqslant 0 \tag{6}$$

Such an optimization problem can be successfully solved with EM algorithm.

Later, [6] introduced the Bayesian approach to extend the PLSA by adding the prior distributions to the and matrices. This approach is called **Latent Dirichlet Allocation (LDA)** and acknowledged the state-of-the-art method. [17] uses the LDA model in their

experiments.

**Additive Regularization of Topic Models (ARTM)** scrutinized in [29], [28] and [26] brought vast diversity into topic modeling techniques.

Still aiming to solve the mentioned earlier optimization problem, ARTM uses regularizers to manage the topic model quality, and, therefore, the optimization problem has the form:

$$L(\Phi, \Theta) + (\Phi, \Theta) \to \max_{\Phi,\Theta} \tag{7}$$

Although there are many useful regularizers, the following three are the most widely used ones:

1. Sparsity $\Phi$: In order to make the $\varphi_{wt}$ distributions more sparse, the sparsity regularizer is used. Having the following form: $R(\Phi) = \sum\limits_{t\in T}\sum\limits_{w\in W} \beta_{wt} \log \varphi_{wt}$, it makes each topic to contain lesser number of tokens from the vocabulary, making them, therefore, more distinct. The only parameter $\beta$ is to be set up.

2. Sparsity $\Theta$: Alike the $\Phi$ sparsity, the $\Theta$ sparsity regularizer ensures each document belongs to fewer topics. Formally, $R(\Theta) = \sum\limits_{d\in D}\sum\limits_{t\in T} \alpha_{td} \log \theta_{td}$.
3. Decorrelation: To make topic even more diverse, thus, increasing the model's value, the regularizer $R = -\frac{\tau}{2}\sum\limits_{t\in T}\sum\limits_{s\in Tt}\sum\limits_{w\in W} \varphi_{wt}\varphi_{ws}$

The regularizers can be combined to acquire more precise and valuable models. Important indicators of the model's quality are $\Phi$ and $\Theta$ sparsity values. However, the interpretability of topics is the most accurate measure, which cannot be estimated analytically, only by assessors.

In addition, ARTM brings new methods in handling the meta data. Modalities are various parts of text besides raw words (e.g. authors, terms, collocations), and their extraction can help in building a better model.

**Multimodal ARTM** solves the following optimization problem with old constraints:

$$\sum_m L_m(\Phi_m, \Theta) + (\cup_m \Phi_m, \Theta) \to \max_{\Phi,\Theta} \tag{8}$$

where $m$ stands for different modalities can be obtained from the texts or their meta data.

Finally, the **Hierarchical ARTM (hARTM)** allows combining two or more distinct ARTM models for each level of the hierarchy, linked with each other. To tailor next-level topics to the topics of the previous level, the regularizer, discussed in [7] is used.

$$R = \tau \sum_{t\in T}\sum_{w\in W} n_{wt} \log \sum_{s\in S} \varphi_{ws}\psi_{st} \tag{9}$$

where $T$ is the number of topics on the previous level, $S$ - number of topics on the next level. $\Psi$ is the stochastic matrix, consisting of probabilities of next-level topics in previous-level ones, which is obtained by solving the following problem:

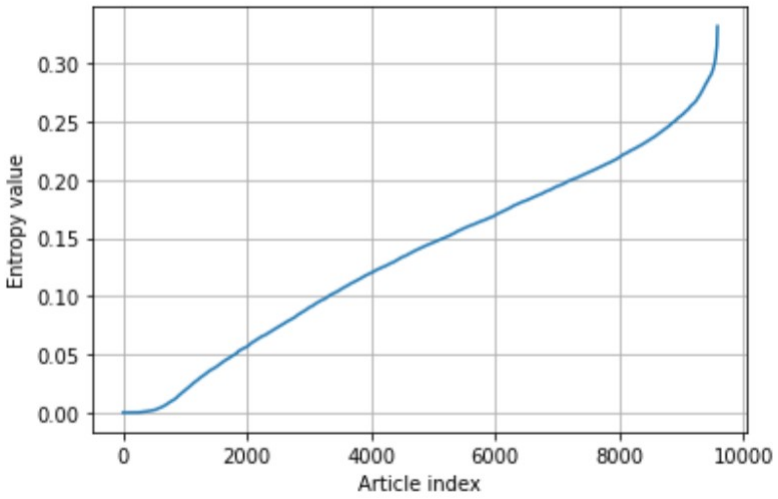


**Figure 1** Hierarchical entropy on a document collection

$$\sum_{t\in T} n_t KL_w \left( \frac{n_w t}{n_t} \,||\, \sum_{s\in S} \varphi_{ws}\psi_{st} \right) \to \max_{\Phi,\Psi} \tag{10}$$

The regularizer ensures that each previous-level topic is the convex sum of the next-level topics, while the $\Psi$ matrix can be valuable in further applications. All regularizers and modalities extensions are applicable to the hARTM models.

In our paper, we compare all of these models in terms of their performance in building reading orders.

### 3.2 Generality

When choosing what to read when starting to explore the novel topic, it is preferable to start with general documents, covering basic terms and ideas of the domain. Measuring generality is the primary problem in reading orders generation.

*Entropy.* We can measure the document generality based on the document's entropy over the topics [17]. The basic intuition behind the entropy is that the higher a document's entropy is, the more topics the document covers hence the more general it is. Using the Shannon entropy, the generality score $g(d)$ of document $d$ is computed as follows:

$$g(d) = -\sum_{t\in T} \theta_{td} log(\theta_{td}) \tag{11}$$

*Hierarchical entropy.* The entropy value has a significant problem that makes it unstable: if the document includes tailored topics from different domains, then its entropy is large, whereas the document is not general. A view of the chart of this entropy on a document collection is shown in Figure 1.

Using hierarchical topic models alleviates this problem. We introduce a hierarchical entropy, that can be calculated using a two-level hierarchical model. Hence, each document $d$ has two embeddings – $\theta_d^1$ and $\theta_d^2$ on first and second level respectively.

Consider the document $d$ is fixed. Let's denote the $h_{st} = -\psi_{st} \times \theta_s^2 d \times \log \theta_s^2 d$ – element of classical entropy of the second-level topic vector multiplied by the probability of the second-level topic $s$ in the first-level topic $t$. Thus, obtain the matrix $H$.

Finally, the hierarchical entropy score is a convolution of the matrix with first-level document embedding:

$$g_h(d) = -\sum_{t\in T}\theta^1_{td}\sum_{s\in S}\psi_{st}\theta^2_{sd}\log\theta^2_{sd} \tag{12}$$

Such score is more stable in terms of above-mentioned example. Its effect is explored in the experiments.

### 3.3 Distance

Distance is the second vital parameter in reading order construction. It defines what documents could be read at the same time independently and what documents are not equal and should be read one after the other.

*Document overlap.* The overlap of two documents can be computed as their weighted Jaccard score [12]. The weighted Jaccard extends the classic Jaccard index, which is defined as the size of the intersection divided by the size of the union of the topic sets assigned to each document, by taking into account their topic probabilities. The overlap score can be computed as follows:

$$o(d_i, d_j) = \frac{\theta^i_{td}\cdot\theta^j_{td}}{|\theta^i_{td}|^2 + |\theta^j_{td}|^2 - \theta^i_{td}\cdot\theta^j_{td}} \tag{13}$$

## 4 Computational Experiment

### 4.1 Data

We use Russian Wikipedia documents and hierarchy of categories to fit and validate our models. Both the Wikipedia texts and categories descriptions are accessible through the Wikimedia Resource. However, the categories hierarchy in Wikipedia is neither the tree nor the set of trees. It is a connected acyclic graph, meaning that from any category we can reach any other.

Since we need the trees only, we use Depth Full Search (DFS) algorithm with limited depth to construct trees fitting our needs.

To validate the model we use the tree with the «Mathematics» as root category and depth 8, containing 9500 documents. On this set, all topic models were fitted and trees validated. Additionally, to compare our results with the initial [17], and explore the scalability of tuned hyperparameters, we gathered the tree with the root of «Machine Learning» and depth 5.

### 4.2 Data processing

Several steps of processing were performed: we lemmatized the documents, all formulas were replaced with special «FORMULA» token.

Moreover, to try the multimodal ARTM, we extracted bigrams from the texts via the TopMine algorithm ( [9]).

### 4.3 Topic models construction

For words from this collection, there were constructed four topic models: LDA, ARTM, PLSA, Hierarchical ARTM. After that, each article was divided into bigrams and monograms and submitted for input to ATM and Hierarchical ARTM. Overall, six topic models were designed with Sparsity $\Phi$ and Sparsity $\theta$ that are shown in Table 1

| Type | Sparsity $\theta$ | Sparsity $\Phi$ for monograms | Sparsity $\Phi$ for bigrams |
|---|---|---|---|
| LDA | 0.76 | 0.93 | - |
| ARTM | 0.77 | 0.93 | - |
| PLSA | 0.74 | 0.93 | - |
| hARTM | 0.87 | 0.95 | - |
| ARTM with bigrams | 0.82 | 0.90 | 0.75 |
| hARTM with bigrams | 0.90 | 0.93 | 0.81 |

**Table 1** Topic models scores

### 4.4 Reading Tree Generation

The tree generation process progressively weaves the reading order for a set of documents by determining the specificity relations among the documents. It takes as input the set of documents, the document-topic assignments $F$, and two parameters: $\tau$ , which defines the minimum overlap between two equivalent documents, and $k$, which defines the maximum difference of their generality scores, $n$, which defines the dynamic decline of $\tau$ during algorithm execution. All parameters are used to define the specificity relations. We use the formulas (11) and (13) for computing document generality and overlap, respectively, but the algorithm is really independent of how document overlap and generality are estimated.

The algorithm builds a complete reading tree in an iterative way [17]. At each round, it handles a subset of similar documents that need to be connected to the tree already created. The algorithm's intention is to grow a sub-tree out of this set of documents and connect it to the existing tree. For this purpose, it first creates the root of this new sub-tree by putting together the most general documents from the set in consideration that are also equivalent. The remaining documents of this set are clustered such that they have some overlap with the root and among them. Each cluster becomes a new set of documents out of which the algorithm will further create new nodes and edges. This process repeats until no more tree growing is possible and there are no documents unprocessed. Initially, the set of documents under consideration contains all documents and the current root node is a dummy node.

The algorithm starts by computing the generality score for each input document $g(d_i)$. All documents are then ordered in descending order of their generality score. A dummy node is created and this node becomes the root of the tree. It is also the first node from where the tree will start to expand (called expansion node). The core operations of the algorithm are described as the function *GetOrder*, which is executed repeatedly but for a different subset of the input documents and growing the tree from a different expansion node each time. Its objective is to build a tree out of its input documents and connect it to the expansion node $v_r$. Steps 1-4 are responsible for the creation of the root node of this tree. This node, $v_s$, groups the more general documents from the current set of documents that have the required topic overlap and generality closeness based on the equivalence condition.

To build this node, the process starts with the most general document $d$. Subsequently, it considers documents in descending order of generality. As long as a document $d_j$ has a close generality score to $d$, and its overlap with all the documents already selected for the node $v_s$ satisfies the equivalence criterion, this document $d_j$ is also added to the set of documents for

**Algorithm 1** Get Order

**procedure** GETORDER(doc set $D$, doc-topic matrix $F$, generality difference threshold $k$, overlap threshold $\tau$, expansion node $v_r$)
  **while** $D \neq 0$
    1. Create a set $S$ containing the most general $d$ in $D$
    2. Select the next most general document $d_j$ in $D$
    **while** $o(d, d_i) > \tau$ and $g(d) - g(d_j) < k$
      **if** $o(d, d_i) > \tau$ **then** add $d_j$ to $S$
      Select the next most general document $d_j$ in $D$
    4. $S$ contains the most general equivalent documents
    and it is mapped to a new node $v_s$
    5. Connect node $v_s$ to the expansion node $v_r$
    6. Remove from $D$ all documents belonging to $S$
    7. $C \leftarrow \{d_j \in D | o(d, d_j) > 0\}$
    8. Divide $C$ into clusters $D_c$ s.t.:
    for each $D_c$ it holds $o(d_i, d_j) > 0, \forall d_i, d_j \in D_c$
    **for all** cluster $D_c$
      GetOrder($D_c, F, k, \frac{\tau}{n}, v_s$)

the node $v_s$. The node $v_s$ becomes the root node for the remaining documents in the current set and it is connected to the node $v_r$ at step 5.

The remaining steps are responsible for re-organizing the rest of the documents in groups such that each group will be used to further grow the tree in a different direction. Step 7 creates a set $C$ of all documents that have non-zero overlap with at least the most general document $d$ of the node $v_s$ just created. The reason is that any document in any node at any level under $v_s$ should have even a distant relation to the documents of $v_s$. Step 8 clusters $C$ into sets $D_c$, such that the documents contained in each set have non-zero overlap among them. The reason is that the nodes of a tree should have some relatedness. The node $v_s$ becomes the new expansion node. The function *GetOrder* is called for each set $D_c$ to grow a new reading tree. Each of these trees is connected to $v_s$.

For simplicity in presentation, Algorithm 1 shows the case of constructing one reading tree. The algorithm builds more than one tree if required. The critical point is the output of step 7. If this is an empty set but there are still documents whose reading order has not been determined, then the algorithm starts a new set of rounds building a new tree that is connected to the dummy root node for the remaining documents.

## 5 Results of Experiments

### 5.1 Machine Learning Catalog of Wikipedia

Due to the small size of this dataset, it could be used for parameter selection. In Table 2 the optimal parameters are shown. There was revealed that optimal parameters depend only on Entropy Type instead of Topic model as we assumed, also parameter $n$ is not mostly useful in the algorithm. Moreover, having optimal parameters $\tau$ and $k$ and changing value of parameter $n$, mostly the quality of the constructed reading order does not depend on $n$ when it is less than one, whereas the greater than one value could have adversely affect on the score, for example, Table 3 shows the dependence of the quality of the constructed reading order with LDA topic model with respect to the $n$ parameter value for optimal parameters $\tau$ and $k$.

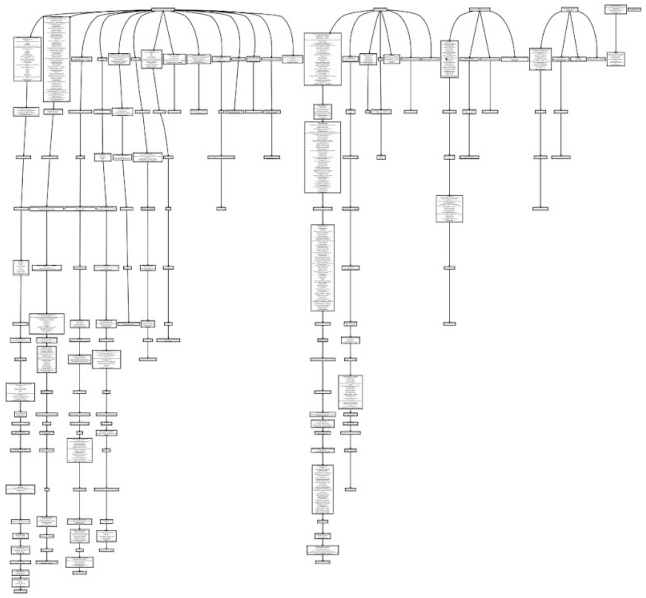

**Figure 2** Reading Order Graph with PLSA topic model

| | Entropy | Hierarchical Entropy |
|---|---|---|
| $\tau$ | 0 | 0.1 |
| $k$ | 0 | 0 |
| $n$ | 1 | 1 |

**Table 2** Optimal parameters

| n | Score |
|---|---|
| 1 | 0.0746 |
| 2 | 0.0942 |
| 3 | 0.0911 |
| 4 | 0.1012 |
| 5 | 0.0917 |
| 6 | 0.0898 |
| 7 | 0.1066 |
| 8 | 0.1012 |
| 9 | 0.1011 |

**Table 3** Scores of reading orders with LDA w.r.t. n

The sample of the reading order graph with PLSA topic model is shown in Figure 2. The results of experiments for reading orders with monograms topic models are shown in Table 4. Comparing with results from [17], where entropy as the measure of document generality and LDA as thy only topic model were used for English Wikipedia hierarchies of ML categories, we get the better results by approximately one order and reveal that applying LDA and entropy give one of the worst scores.

| Topic model | Score | Execution time (sec) |
|---|---|---|
| LDA | 0.0746 | 7.3 |
| hARTM with Hierarchical Entropy | 0.0745 | 6.8 |
| ARTM | 0.0745 | 7.2 |
| hARTM | 0.0744 | 9.3 |
| PLSA | 0.0741 | 15.2 |

**Table 4** Results for Machine Learning Catalog of Wikipedia

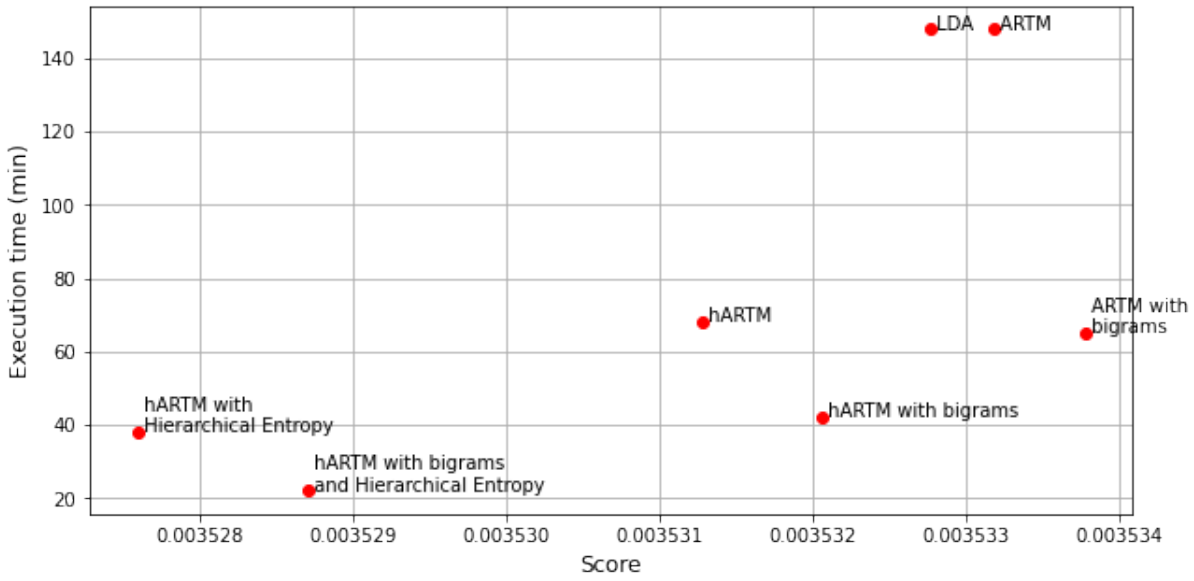


**Figure 3** Results for Mathematics Catalog of Wikipedia

### 5.2 Mathematics Catalog of Wikipedia

We tested how our approach and optimal parameters for $\tau$, $k$, $n$ scales with the number of documents. This dataset contains many more articles than the previous one and it was decided to apply bigrams selection and topic models that used them. In Table 5 the results of experiments for reading orders with optimal parameters and different topic models are presented. Unfortunately, we couldn't wait for the algorithm to finish working with the PLSA model and optimal parameters, we estimated the results according to execution times and scores with this topic model for fewer document collections.

| Topic model | Score | Execution time |
|---|---|---|
| ARTM with bigrams | 0.003534 | 1h 5min |
| ARTM | 0.003533 | 2h 28min |
| LDA | 0.003533 | 2h 28min |
| hARTM with bigrams | 0.003532 | 42min |
| hARTM | 0.003531 | 1h 8min |
| hARTM with bigrams and Hierarchical Entropy | 0.003529 | 22min |
| hARTM with Hierarchical Entropy | 0.003528 | 38min |
| PLSA (estimation) | 0.003510 | $3{\cdot}10^8 min$ |

**Table 5** Results for Mathematics Catalog of Wikipedia

## 6 Conclusion

Overall, as a result of our experiments, we came to the following conclusions: there is scalability for optimal parameters of the algorithm with different topic models. Moreover, it is unnecessary to find the optimal parameters each time you change the topic model of the algorithm. They depend only on the type of entropy that you use. However, scaling does not

save the rating of reading orders with different topic models. Although, you can use that reading orders with Hierarchical Entropy has shown better results in both execution time and score, whereas bigrams selection worsens the score of reading order a little bit but reduces execution time in about twice instead.

## References


[1] Meysam Asgari-Chenaghlu. Word vector representation, word2vec, glove, and many more explained. 02 2017.

[2] Anton Belyy. Construction and quality evaluation of heterogeneous hierarchical topic models. *CoRR*, abs/1811.02820, 2018.

[3] Samy Bengio, Jason Weston, and David Grangier. Label embedding trees for large multi-class tasks. In John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta, editors, *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, pages 163–171. Curran Associates, Inc, 2010.

[4] David M. Blei, Thomas L. Griffiths, and Michael I. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *J. ACM*, 57(2):7:1–7:30, 2010.

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[6] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[7] N.A. Chirkova. Additive regularization for hierarchical multimodal topic modeling. *Machine Learning and Data Analysis*, 2:187–200, 01 2016.

[8] Albi Dode and Silvester Hasani. Pagerank algorithm. *10.9790/0661-1901030107*, 19:2278–661, 02 2017.

[9] Ahmed El-Kishky, Yanglei Song, Chi Wang, Clare Voss, and Jiawei Han. Scalable topical phrase mining from text corpora. *Proceedings of the VLDB Endowment*, 8, 06 2014.

[10] M. A. Eremeev and Konstantin Vorontsov. Lexical quantile-based text complexity measure. In *RANLP*, 2019.

[11] Ao Feng and James Allan. Incident threading for news passages. In David Wai-Lok Cheung, Il-Yeol Song, Wesley W. Chu, Xiaohua Hu, and Jimmy J. Lin, editors, *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 1307–1316. ACM, 2009.

[12] Gregory Grefenstette. *Explorations in automatic thesaurus discovery.* Kluwer Academic Publishers, Dordrecht, NL, 1994.

[13] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc., 1999.

[14] James G. Jardine. Automatically generating reading lists. Technical Report UCAM-CL-TR-848, University of Cambridge, Computer Laboratory, February 2014.

[15] T. Jiang, L. Wang, and K. Zhang. Alignment of trees: an alternative to tree edit. In M. Crochemore and D. Gusfield, editors, *Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching*, number 807 in Lecture Notes in Computer Science, pages 75–86, Asilomar, CA, 1994. Springer-Verlag, Berlin.

[16] Bahar Karaoğlan and Tarik Kisla. Course prescription using ontologies. pages 184–186, 05 2013.

[17] Georgia Koutrika, Lei Liu, and Steven J. Simske. Generating reading orders over document collections. In Johannes Gehrke, Wolfgang Lehner, Kyuseok Shim, Sang Kyun Cha, and Guy M. Lohman, editors, *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, pages 507–518. IEEE Computer Society, 2015.

[18] Lei Liu, Prakash Mandayam Comar, Sabyasachi Saha, Pang-Ning Tan, and Antonio Nucci. Recursive nmf: Efficient label tree learning for large multi-class problems. In *ICPR*, pages 2148–2151. IEEE Computer Society, 2012.

[19] Lei Liu and Pang-Ning Tan. A framework for co-classification of articles and users in wikipedia. In Jimmy Xiangji Huang, Irwin King, Vijay V. Raghavan 0001, and Stefan Rueger, editors, *2010 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2010, Toronto, Canada, August 31 - September 3, 2010, Main Conference Proceedings*, pages 212–215. IEEE Computer Society, 2010.

[20] Gary Marchionini. Exploratory search: From finding to understanding. *Commun. ACM*, 49(4):41–46, 2006.

[21] David Mimno, Wei Li, and Andrew McCallum. Mixtures of hierarchical topics with pachinko allocation.

[22] Roberto Pirrone, Giovanni Pilato, Riccardo Rizzo, and Giuseppe Russo. G.: Learning path generation by domain ontology transformation. volume 3673, pages 359–369, 09 2005.

[23] Dafna Shahaf and Carlos Guestrin. Connecting two (or less) dots: Discovering structure in news articles. *TKDD*, 5(4):24:1–24:31, 2012.

[24] Valery Solovyev, Vladimir Ivanov, and Marina Solnyshkina. Assessment of reading difficulty levels in russian academic texts: Approaches and metrics. *Journal of Intelligent and Fuzzy Systems*, 34:1–10, 04 2018.

[25] E. Tanaka and K. Tanaka. The tree-to-tree editing problem. *International Journal of Pattern Recognition and Artificial Intelligence*, 2:221–240, 1988.

[26] K. V. Vorontsov and A. A. Potapenko. Additive regularization of topic models. *Machine Learning, Special Issue on Data Analysis and Intelligent Optimization with Applications*, 101(1):303–323, 2015.

[27] Konstantin Vorontsov, Oleksandr Frei, Murat Apishev, Peter Romov, and Marina Dudarenko. Bigartm: open source library for regularized multimodal topic modeling of large collections. In *International Conference on Analysis of Images, Social Networks and Texts*, pages 370–381. Springer, 2015.

[28] Konstantin Vorontsov, Oleksandr Frei, Murat Apishev, Peter Romov, Marina Suvorova, and Anastasia Yanina. Non-bayesian additive regularization for multimodal topic modeling of large collections. 10 2015.

[29] Konstantin Vorontsov and Anna Potapenko. Additive regularization of topic models. *Machine Learning*, 101(1-3):303–323, 2015.

[30] Ryen W. White and Resa A. Roth. *Exploratory Search: Beyond the Query-Response Paradigm.* Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan and Claypool Publishers, 2009.

[31] Elias Zavitsanos, Georgios Paliouras, and George A. Vouros. Non-parametric estimation of topic hierarchies from texts with hierarchical dirichlet processes. *J. Mach. Learn. Res*, 12:2749–2775, 2011.