BATS: A Spectral Biclustering Approach to Single Document Topic Modeling and Segmentation

SIRUI WANG, Department of Computer Science, College of William and Mary
YUWEI TU, Zoomi Inc.
QIONG WU, Department of Computer Science, College of William and Mary
ADAM HARE, Zoomi Inc.
ZHENMING LIU, Department of Computer Science, College of William and Mary
CHRISTOPHER G. BRINTON, School of Electrical and Computer Engineering, Purdue University
YANHUA LI, Department of Computer Science, Worchester Polytechnic Institute

Existing topic modeling and text segmentation methodologies generally require large datasets for training, limiting their capabilities when only small collections of text are available. In this work, we reexamine the inter-related problems of “topic identification" and “text segmentation" for sparse document learning, when there is a single new text of interest. In developing a methodology to handle single documents, we face two major challenges. First is *sparse information*: with access to only one document, we cannot train traditional topic models or deep learning algorithms. Second is *significant noise*: a considerable portion of words in any single document will produce only noise and not help discern topics or segments. To tackle these issues, we design an unsupervised, computationally efficient methodology called BATS: Biclustering Approach to Topic modeling and Segmentation. BATS leverages three key ideas to simultaneously identify topics and segment text: (i) a new mechanism that uses word order information to reduce sample complexity, (ii) a statistically sound graph-based biclustering technique that identifies latent structures of words and sentences, and (iii) a collection of effective heuristics that remove noise words and award important words to further improve performance. Experiments on four datasets show that our approach outperforms several state-of-the-art baselines when considering topic coherence, topic diversity, segmentation, and runtime comparison metrics.

Additional Key Words and Phrases: Biclustering, Topic modeling, Text segmentation

ACM Reference Format:
Sirui Wang, Yuwei Tu, Qiong Wu, Adam Hare, Zhenming Liu, Christopher G. Brinton, and Yanhua Li. 2020. BATS: A Spectral Biclustering Approach to Single Document Topic Modeling and Segmentation. 1, 1 (August 2020), 22 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Authors' addresses: Sirui Wang, swang23@cs.wm.edu, Department of Computer Science, College of William and Mary, 200 Stadium Dr, Williamsburg, Virginia, 23185; Yuwei Tu, yuwei.tu@zoomi.ai, Zoomi Inc., 640 Lee Road, Wayne, Pennsylvania, 19087; Qiong Wu, qwu05@cs.wm.edu, Department of Computer Science, College of William and Mary, 200 Stadium Dr, Williamsburg, Virginia, 23185; Adam Hare, adam.hare@zoomi.ai, Zoomi Inc., 640 Lee Road, Wayne, Pennsylvania, 19087; Zhenming Liu, zliu05@cs.wm.edu, Department of Computer Science, College of William and Mary, 200 Stadium Dr, Williamsburg, Virginia, 23185; Christopher G. Brinton, cgb@purdue.edu, School of Electrical and Computer Engineering, Purdue University, 610 Purdue Mall, West Lafayette, Indiana, 47907; Yanhua Li, yli15@wpi.edu, Department of Computer Science, Worchester Polytechnic Institute, 100 Institute Rd, Worcester, Massachusetts, 01609.

XXXX-XXXX/2020/8-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Innovations in topic identification and text segmentation have demonstrated the potential for automated analyses of large collections of documents. Broadly speaking, *topic identification* refers to finding a collection of topics (e.g., groups of words) that represent a given document, whereas *document segmentation* refers to partitioning a document into components (e.g., sentences) about the topics. Existing solutions to these problems are usually based on analyzing statistical patterns in text across datasets that consist of large collections of documents. For example, the popular Latent Dirichlet Allocation (LDA) algorithm for topic modeling [3] assumes that each document comprising a corpus, and every word in them, are generated according to the latent Dirichlet process. With this assumption, EM-based algorithms can then be employed to infer the latent states of the documents [24]. Word embedding models such as word2vec [31] and GloVe [36] have also become popular, building joint distributions of word sequences by transforming every word in a document into a high-dimensional space learnt over a large corpus. The resulting high-dimensional representations then help to identify topics in the document and perform segmentation based on these topics.

While algorithms for finding topics [3, 14, 24] and segmenting documents [10, 23, 41] have been extensively studied, none have fully addressed the “new and single document” issue. In this setting, we may need to analyze a newly created text whose topics have not been seen before. Such cases are especially prevalent in politics, when new names of political actors or nicknames for events (e.g. “Brexit” or scandals ending in “gate”) may appear suddenly and require rapid analysis. Neologisms are not the only problem; existing words may acquire new context-specific meaning. For instance, the word “like” has acquired meaning in the age of social media (i.e., due to “like buttons”) it lacked before the rise of Facebook and other such platforms [6]. Academia is subject to the same problem when new articles/books/lectures appear without enough training data for thorough analysis.[1] Any model operating on user-generated text data will eventually be presented with content containing topics it has not seen previously. Furthermore, this new content may be the most salient, as it is likely to reflect evolving events or trends which users are most interested in exploring. Rapid analysis to identify such topics and segments is often necessary while the content is most relevant (for instance, the night of an election) [26].

Existing deep and statistical learning approaches are unsuitable solutions for these situations. For one, pre-trained models by nature will rely on large amounts of historical data [50], and thus it is often difficult to adapt them effectively for these situations. Relying on existing word embeddings is equally difficult, as even powerful contextualized embedding models [15, 30] struggle to capture new word senses emerging from developing events that have limited training data. Moreover, training new models (i.e., on the newly relevant, emerging data) is error-prone and costly, because the new dataset may not be sufficiently large to produce a generalized model and the cost of frequent training/re-training may be prohibitive for real-time systems [15].

1.1 BATS: Objectives and Key Techniques

In this paper, to address the challenges outlined above, we design a statistically sound, computationally efficient, unsupervised algorithm that can simultaneously extract topics and segment text from a single document of interest. Designing such an algorithm is challenging because we need to determine model parameters on a sparse dataset. Our development is guided by three key ideas:

1.1.1 Idea 1: Using word ordering information properly. Traditional topic modeling approaches assume bag-of-words models [3] where information on the order in which words appear is neglected.

[1]Consider, for example, the introduction of a technique with a name which previously had no meaning, or the changes in the use of the word “transformer" in the scientific community before and after the publication of [45].

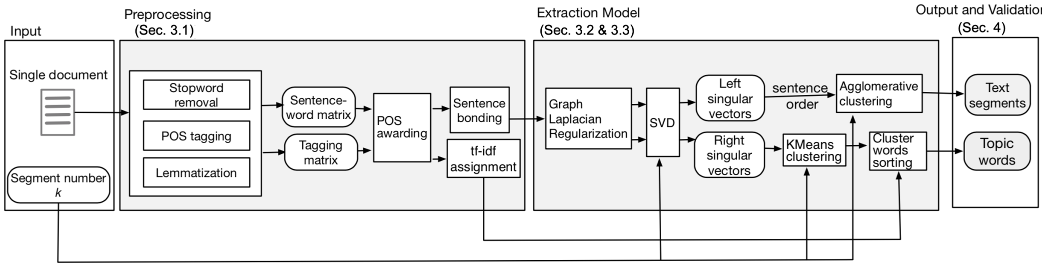


Fig. 1. Block diagram summary of the modules comprising BATS, the spectral biclustering methodology we develop in this paper for single-document topic modeling and text segmentation.

While this has proven effective in the analysis of full corpuses, compression to a bag-of-words in the case of a single document may lose information valuable to the task at hand. The recent success of recurrent models and the addition of positional encodings in non-recurrent models for the application of machine translation [45] is further evidence of the potential value of word-order information on single document.

Motivated by this, our approach aims to leverage word-order information to achieve good performance in the presence of a small, single document training dataset. In particular, we consider the location of words in neighboring sentences. In designing this mechanism, we will make two assumptions guided by basic rules of written language: (i) words appearing in the same sentences are more likely to be on the same topic, and (ii) words located in nearby sentences are more likely to be on the same topic.

1.1.2 Idea 2: Design a biclustering algorithm that addresses sparsity. For joint topic modeling and text segmentation, we will find it convenient to model documents with sentence-word matrices. But word-to-word interactions and word-to-sentence interactions are noisy by nature [5]. This problem becomes even more pronounced with small datasets like single documents where these interactions are likely to be sparse (e.g., the sentence-word matrices for datasets considered in this paper have only 15% of entries nonzero on average). A well-designed denoising process is necessary so that a sentence-word matrix can be utilized effectively in the downstream topic extraction and text segmentation tasks.

Our approach connects the denoising problem here with the denoising problem in stochastic block models [38]. In particular, we design a specialized spectral biclustering algorithm which operates on a regularized sentence-word graph Laplacian representation of a document to address sparsity. The topics and segments emerge from clustering the right and left singular vectors of this Laplacian. Given this, we term our overall solution **BATS**: **B**iclustering **A**pproach for **T**opic modeling and **S**egmentation.

1.1.3 Idea 3: Optimize heuristics to analyze single documents. We design a number of heuristics to enhance our algorithm's performance. Our heuristics are designed based on two major observations: (i) extremely low frequency words (i.e. words with only one appearance in the text) tend to introduce noise to document analysis and thus need to be removed, and (ii) part-of-speech (POS) tagging can help to identify more important elements of a document and thus should be considered in our model. Therefore, we remove the low frequency words in the text, but award the important words according to their POS tags. Specifically, because nouns and verbs often convey the body and condition of a sentence, they are typically more informative in topic modeling than other parts of speech [21]. As a result, we award nouns and verbs by giving them additional weight.

1.1.4 Experimental validation. We evaluate BATS against five baselines in both topic modeling and text segmentation tasks. For topic modeling, we compare performance in terms of topic coherence (i.e., quality of individual topics) and topic diversity (i.e., overlap in topic words) on three datasets, in which we find that BATS always performs the best in topic diversity, performs comparably to the best existing algorithms in topic coherence, and obtains the best with respect to a composite metric. For text segmentation, we add in one more standard dataset, and show that we outperform baselines substantially in most cases in terms of agreement with a ground truth. We also show that our method scales well with the size of the input document compared with the baselines.

1.2 BATS: Architecture and Roadmap

Figure 1 outlines the methodology we develop and provides a roadmap for the paper. The inputs to BATS are a single document and a single hyperparameter (segment number, which also indicates topic number). Then, the two major stages of BATS are preprocessing and extraction. In the preprocessing stage (Sections 3.1 and 3.2), we leverage ideas 1 and 3 to build an effective feature matrix representation of a document under sparse and noisy conditions. In the extraction stage (Sections 3.2 and 3.3), we use idea 2 to identify low-dimensional representations of the signals through spectral biclustering, with agglomerative methods to segment the text and KMeans to identify the topics. Our subsequent evaluation (Section 4) assesses performance of the resulting text segments and topic words in terms of diversity, coherence, and segmentation metrics.

1.3 Summary of Contributions

Our key contributions are summarized as follows:

- We develop a novel methodology called BATS that performs topic modeling and text segmentation on a single document simultaneously. BATS is unsupervised and scalable in its implementation, as it does not rely on pre-trained word embedding models.
- We connect the joint topic extraction and segmentation problem to spectral biclustering of sentence-word matrices, and show how a factorization of the graph Laplacian with appropriate pre-processing and post-clustering can lead to effective results.
- Our evaluation on several datasets establishes that BATS achieves higher performance on topic modeling and text segmentation metrics when compared with key baselines on single documents, and shows that BATS is scalable with document length.

2 RELATED WORK

We identify three areas of related work: biclustering techniques, topic modeling, and text segmentation algorithms.

2.1 Biclustering Techniques

Biclustering techniques (e.g., [16, 17, 43]) have been proposed to model interactions among two types of nodes represented in a bipartite graph, with nodes of each type grouped into clusters according to different methods. These techniques are widely used in part because of their sound theoretical properties [17]. In [16] and [25], the authors propose algorithms which translate input data into bipartite graphs and apply spectral techniques to the adjacency matrices; in [16], a block diagonal structure is assumed, while in [25], the case of a checkerboard pattern is considered, with implications to the spectral decomposition. [43] can be viewed as an extension of the algorithm in [16] to deal with asymmetric data matrices. By contrast, [17] proposes a probabilistic approach to graph biclustering, where the input data matrix is treated as a joint probability distribution between two random variables, which are then clustered according to relative entropy and mutual information metrics. Our work builds off the spectral clustering foundations in [16, 43], accommodating rectangular sentence-word data matrices instead of traditionally assumed square matrices.

2.2 Topic Modeling

Several models have been proposed to extract topics from a corpus consisting of multiple long documents, including Latent Semantic Analysis (LSA) [14], Non-negative Matrix Factorization (NMF) [35], Probabilistic Latent Semantic Analysis (pLSA) [24], Latent Dirichlet Allocation (LDA) [3], and variants on LDA, e.g., hierachical modeling [44] (see [12] for a survey). Analysis on short texts, however, usually faces the issue of sparsity in word occurrences. To overcome this challenge, works such as [48, 49] make additional assumptions on word co-occurrence patterns; [34, 51] have resorted to word embeddings which leverage pre-trained models; [11, 20] depend on further external knowledge including social relationships in microblogs and user preferences.

Different from these methods, ours aims at identifying topics in a single, newly created document without an extensive training/re-training component. To overcome issues of input data sparsity and noise, BATS turns to word-ordering information between sentences and regularization in the spectral clustering phase, as opposed to making additional assumptions on word co-occurrence patterns. Through evaluation on several datasets, we show that BATS outperforms these methods on single document topic modeling in terms of topic coherence, topic diversity, and scalability metrics.

2.3 Text Segmentation

Text segmentation algorithms are designed to detect breakpoints in a document and split the document into multiple segments accordingly. Algorithms such as Lexical Chains [28] and TextTiling [23] use lexical co-occurrence and distribution patterns to divide sets of paragraphs into multi-paragraph sub-blocks that become segments. A potential drawback of these approaches, however, is that the segments are not associated or labeled with explicit topic information, and that it is not always clear how to translate from a lexical distribution to topics. This motivates the consideration of topic modeling and text segmentation jointly.

More recently, to improve segmentation performance, topic-based segmentation methods such as TopSeg [4], LDA_MDP [33], and TopicTiling [41] have been proposed. Similar to the topic modeling algorithms discussed above, these segmentation methods depend heavily on the training process, and usually require training on a large corpus [8]. This is problematic when only small datasets are available, let alone the single document case that we consider in this paper. Through biclustering of the sentence-word matrix and development of other pre-processing techniques, BATS does not demand an expensive training process. Further, our evaluation shows that BATS outperforms the segmentation methods discussed here on single documents across several datasets in terms of standard segmentation metrics.

3 SPECTRAL BICLUSTERING METHODOLOGY

As shown in Figure 1, our proposed methodology BATS consists of two main stages: the text preprocessing stage (Section 3.1) and the extraction stage, with the latter broken down into graph Laplacian regularization (Section 3.2) and sentence/word clustering (Section 3.3). Topics and segments emerge from the word and sentence clusters, respectively. In this section, we detail the development of these modules.

3.1 Document Preprocessing and Matrix Construction

Consider an input document comprised of m sentences, indexed $i = 1, \ldots, m$. We denote $\mathcal{W} = \{w_1, \ldots, w_n\}$ as the set of words we are interested in for modeling, indexed $j = 1, \ldots, n$. In defining $\mathcal{W}$, we do not include all the words that ever appear in the document; instead, a word is included in $\mathcal{W}$ if and only if it appears in more than one sentence in the document and it is not in a stopword list.[2] In this way, $\mathcal{W}$ excludes "degree-one" words that can skew models in single documents; we observe that these words often behave as pure noise in our inference algorithms.

Let $X = [X_{ij}] \in \mathbb{R}^{m \times n}$ denote the sentence-word matrix. We develop two steps to construct X, taking into account both word order and parts-of-speech information:

3.1.1 Step 1. Using parts-of-speech information. Our first optimization trick is based on parts-of-speech (POS) tags, which are generated through analysis of the word positions in the sentences [21]. In particular, the lexical model presented in [18] shows hierarchies exist according to the syntactic and semantic similarities of the words; looking into the hierarchies, it is clear that nouns and verbs convey more information than other word types, and thus should be given a larger weight [40]. As a result, letting $X^o = [X^o_{ij}]$ where X^o_{ij} is the number of occurrences of word $w_j \in \mathcal{W}$ in sentence i, we define

$$X^a = X^o + \lambda T, \tag{1}$$

where $T = [T_{ij}]$, $T_{ij} = 1$ if $X^o_{ij} \neq 0$ and w_j is tagged as a noun or verb in sentence i, and $T_{ij} = 0$ otherwise. $\lambda > 0$ is a scalar parameter for awarding POS; by default, $\lambda = 1$. In our implementation, Python's `spaCy` module is used to tag the words, as this pre-trained model based on word positions is more robust to novel words or topics than would be, for instance, a word-embedding model.

3.1.2 Step 2. Transformation by using word-order information. Our incorporation of word-order information is based on the intuition that words in neighboring sentences are likely to be similar in their constituent topics, with this effect decaying as the sentences grow further apart. Assumptions on words appearing within a certain window being related can be found in other text analysis techniques as well, including word embedding models [36]. Concretely, we bond neighboring sentences to the current sentence according to

$$X_i = \sum_{\ell=i-w}^{i+w} d^{|\ell-i|} X^a_\ell, \qquad i = 1, \ldots, m, \tag{2}$$

where $X_i = (X_{i1}, \ldots, X_{in})$ is the ith row of X and X^a_ℓ is the ℓth row of X^a for $\ell = 1, \ldots, m$ (for $\ell < 1$ and $\ell > m$, X^a_ℓ is taken as a vector of zeros). Parameter w controls the size of the bonding window, and $d \in [0, 1]$ is a decay rate for the distance. In this way, the presence of a word in one sentence will impact neighboring sentences, and words appearing in several consecutive sentences are increased in importance. Doing so also alleviates the issue of sparsity associated with single documents, as each sentence's data smoothens its neighbors' representations too. The procedure for tuning w and d will be discussed in Section 3.2.

3.2 Graph Laplacian and Singular Vectors

Consider the bipartite graph $\mathcal{G}(X)$ of the sentence-word matrix X, where the sentences $i = 1, \ldots, m$ and words $j = 1, \ldots, n$ each form a node set, and edge (i, j) of weight $X_{i,j}$ is in $\mathcal{G}(X)$ if and only if $X_{i,j} \neq 0$. In this section, we derive a graph Laplacian for X according to $\mathcal{G}(X)$, and employ that to construct low dimensional embeddings. Graph Laplacians have been noted for their success in

[2]Our stopword list combines the English and Spanish lists from the NLTK module: `https://www.nltk.org/book/ch02.html`.

spectral clustering algorithms [46], which we will develop in Section 3.3. Specifically, our approach here consists of two steps:

3.2.1 Step 1. Regularizing the graph Laplacian. Given the issue of document sparsity, and the asymmetric nature of the bipartite graph, we derive a regularized version of the graph Laplacian as in [1, 9, 38]. Formally, define two diagonal matrices $P = \mathrm{diag}(P_1, ..., P_n) \in \mathbb{R}^{n\times n}$ and $O = \mathrm{diag}(O_1, ..., O_m) \in \mathbb{R}^{m\times m}$ where $P_j = \sum_{i=1}^m X_{ij}, j = 1, ..., n$ and $O_i = \sum_{j=1}^n X_{ij}, i = 1, ..., m$ are the row and column sums of X. With regularization parameters $\tau_p, \tau_o \geq 0$, the regularized graph Laplacian $L \in \mathbb{R}^{m\times n}$ is computed as

$$\begin{aligned} P_\tau &= P + \tau_p I_p, \\ O_\tau &= O + \tau_o I_o, \\ L &= (O_\tau)^{-\frac{1}{2}} X (P_\tau)^{-\frac{1}{2}}, \end{aligned} \tag{3}$$

where I_p and I_o are identity matrices. Multiplying these by regularization parameters τ_p and τ_o can resolve issues due to poor concentration since the degrees for every vertex are inflated. Following prior work [38] which has indicated that such regularization parameters should be proportional to the average degrees of the vertices (so that the asymptotic bounds will be indicative of the mis-clustering rate), we set the average degrees as defaults, i.e., $\tau_p = \sum_j P_j/n$ and $\tau_o = \sum_i O_i/m$.

3.2.2 Step 2. Obtaining a low dimensional embedding. We consider the singular value decomposition (SVD) of L. By definition, the SVD yields

$$L = U\Sigma V^T, \tag{4}$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are unitary matrices and Σ contains the singular values $\sigma_1, ..., \sigma_{\max\{m,n\}}$ on its diagonal. Since $L^T L = V(\Sigma^T\Sigma)V^T$ is a measure of similarity between words, counting their degrees of connectivity via sentences, and $LL^T = U(\Sigma\Sigma^T)U^T$ is a measure of similarity between sentences, counting their degree of connectivity via words, we expect the SVD can be used to cluster words (using V) and sentences (using U). Further, as the eigenvalues of $L^T L$ and LL^T are the square of the singular values in Σ, we introduce another parameter k which denotes the number of left $U_1, ..., U_k \in \mathbb{R}^m$ and right $V_1, ..., V_k \in \mathbb{R}^n$ dominant singular vectors used, where we assume the singular values are in decreasing order $\sigma_1 \geq \sigma_2 \geq \cdots$. We then re-normalize the rows of the resulting matrices

$$V' = [V'_{i\ell}] = [V'_1\ V'_2\ \cdots\ V'_k], \quad U' = [U'_{j\ell}] = [U'_1\ U'_2\ \cdots\ U'_k] \tag{5}$$

to have unit length, i.e., so that $\sum_\ell V'^2_{i\ell} = \sum_\ell U'^2_{j\ell} = 1$ for each sentence i and word j. Following [46], which suggests that the dimensionality should be consistent with the number of clusters to be grouped, we use the same parameter k for both U and V.

The full matrix decomposition process developed in Sections 3.1 and 3.2 is summarized in Algorithm 1.

3.2.3 Impact of w and d. Recall the window w and decay d parameters from (2). We investigate the impact of these parameters on the matrix decomposition in (5) by considering the L2-norm distances between the resulting sentence vectors in U. Figure 2 gives heatmaps of these distances for an arbitrary document in one of our datasets (see Section 4.1), where entry (x, y) on each heatmap takes the value $\sum_\ell (U_{x\ell} - U_{y\ell})^2$. Since neighboring sentences should cover similar topics, we seek values of w and d for which ordering information is clearly embedded in the matrix. In Figure 2(a), for small values of w (i.e., $w = 0, 1$), the sentence order is less clear as the elements near the diagonal are more blurry. As w increases, the pattern becomes more obvious, and when $w = 3$ we observe clear block patterns in the heatmap. When w is increased further (i.e., to $w = 5$),

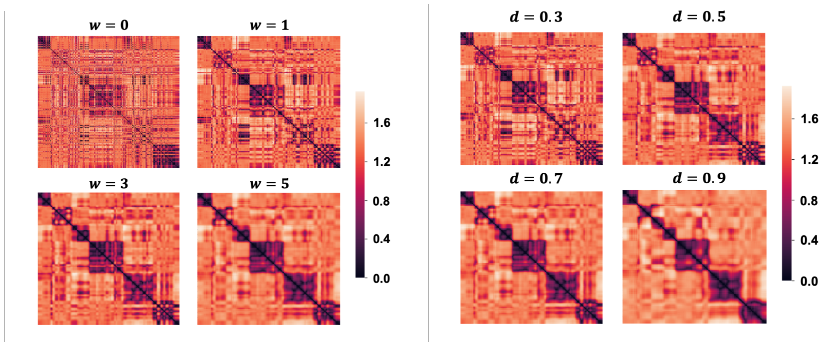
Algorithm 1 Matrix decomposition on regularized Laplacian.

INPUT: Original sentence-word matrix X^o, POS-based matrix T
PARAMETER: Awarding value λ, window size w, decaying rate d, segment number k
OUTPUT: Matrix U for sentences and V^T for words

function MAT_DECOMP(X^o, w, d)
  if $\lambda > 0$ **then**
    $X^a = X^o + \lambda T$ //Word awarding
  else
    $X^a = X^o$
  $F \leftarrow$ tf-idf(X^a) //Tf-idf assignment
  for $i \leftarrow 1, \ldots, n$ **do**
    $X_i = \sum_{\ell=i-w}^{i+w} d^{|\ell-i|} X_\ell^a$ //Sentence bonding
  for $j \leftarrow 1, \ldots, n$ **do**
    $P_j \leftarrow \sum_{i=1}^{m} X_{ij}$
  for $i \leftarrow 1, \ldots, m$ **do**
    $O_i \leftarrow \sum_{j=1}^{n} X_{ij}$
  $\tau_p \leftarrow \sum_j P_j/n$, $P_\tau \leftarrow P + \tau_p I_p$ //Regularization
  $\tau_o \leftarrow \sum_i O_i/m$, $O_\tau \leftarrow O + \tau_o I_o$ //Regularization
  $L = (O_\tau)^{-\frac{1}{2}} X (P_\tau)^{-\frac{1}{2}}$ //Graph Laplacian
  $U\Sigma V^T = L$ //Singular value decomposition
  for $u' \leftarrow$ rows of U **do**
    $u' \leftarrow u'[1:k]$ //Reserve first k dimensions
    $u' \leftarrow u' / \sqrt{\sum_i u_i'^2}$ //L2 normalization on U'
  for $v' \leftarrow$ rows of V **do**
    $v' \leftarrow v'[1:k]$ //Reserve first k dimensions
    $v' \leftarrow v' / \sqrt{\sum_i v_i'^2}$ //L2 normalization on V'
  return U', V', F //U' for sentences, V' for words

the sharpness of the block pattern does not continue to improve; intuitively, sentences at the far ends of the bonding window for large w will have higher dissimilarity, but this effect is blunted by the decay d (which is 0.7 here). Since a higher w also increases the runtime of the method, in considering several documents, we find that the best choice of w is typically between 3 and 5 (i.e., the number of topic-neighboring sentences is 6 to 10).

By this logic, then, the value of d should be significantly lower than 1. As it is decreased in Figure 2(b) (i.e., from $d = 0.9$), we see that the sharpness of the blocks improves, with $d = 0.7$ giving the clearest pattern. Beyond this (i.e, $d = 0.5, 0.3$), however, the sharpness begins to decrease. In these cases, neighboring sentences are assigned lower weights, confirming that the SVD uncovers topic similarity between neighbors. In considering several documents, we find that the best choice is $d \approx 0.7$ for this reason.



(a) Varying w (b) Varying d

Fig. 2. Heatmaps of the pairwise distances between sentence vectors in the SVD for a sample document taken from the Introductions dataset (see Section 4.1) under different values of parameters w and d. (a) varies w for fixed $d = 0.7$, and (b) varies d for fixed $w = 3$. Block similarity is the clearest for the combination $w = 3, d = 0.7$. Other documents yield qualitatively similar results.

3.3 Word and Sentence Clustering

With the embedding from (5) in hand, we move to obtain topics and segments via spectral clustering of the word V and sentence U matrices respectively. To justify that clustering can provide desired results in both cases, we consider the problem from a graph cut point of view [46], where the cuts are taken on a similarity graph of words or sentences.

Formally, let $G = [g_{i,j}] \in \mathbb{R}^{n\times n}$ be the similarity matrix among a set of n nodes $v_1, ..., v_n$ (i.e., words or sentences), where $g_{i,j} \geq 0$ is the similarity between nodes v_i and v_j. We seek to minimize $KCut(S_1, ..., S_k) = \frac{1}{2}\sum_{p=1}^{k} cut(S_p, \bar{S_p})$ while $cut(S_p, \bar{S_p}) = \sum_{\substack{i\in S_p,\\ j\in \bar{S_p}}} g_{i,j}$. Note $S = \{S_1, S_2, ..., S_k\}$ is a grouping of the nodes into k disjoint sets $S_1, ..., S_k$. A simple and straight forward solution for this minimization problem is to cut off individual nodes which are weakly connected to the rest. However, there is usually no topic with one word or text segment with one sentence, therefore, the groups of words or sentences are supposed to have more balanced sizes. As a result, the objective function needs to take group sizes into consideration and build the balanced cut problem

$$BCut(S_1, ..., S_k) = \sum_{p=1}^{k} \frac{cut(S_p, \bar{S_p})}{|S_p|}. \tag{6}$$

Taking group sizes into account makes the problem NP hard and requires further relaxation. We reorganise the problem by defining a group indication matrix $H = [h_1 \cdots h_k] \in \mathbb{R}^{n\times k}$ consisting of k weighted indicator vectors $h_p = (h_{1,p}, ..., h_{n,p})^T, p = 1, ..., k$ where

$$h_{i,p} = \begin{cases} 1/\sqrt{|S_p|} & \text{if node } v_i \in S_p \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

We can see $H^T H = I$ where I is the identity. For the node similarity graph G, we have its degree matrix as $D = \mathrm{diag}(d_1, ..., d_n)$ where $d_i = \sum_j g_{ij}, i = 1, ..., n$ and the unnormalized graph Laplacian as $L_u = D - G$. Through some easy math, we can get

$$h_p^T L_u h_p = \frac{cut(S_p, \bar{S_p})}{|S_p|}, \text{ for } p = 1, ..., k. \tag{8}$$

Combining this with (6), we conclude that

$$BCut(S_1, ..., S_k) = \sum_{p=1}^{k} h_p^T L_u h_p = Tr(H^T L_u H). \tag{9}$$

Thus, the minimization problem can be presented as

$$\min_{S_1,\cdots,S_k} Tr(H^T L_u H) \text{ subject to } H^T H = I, H \text{ defined as (7).} \tag{10}$$

This problem is equivalent to minimizing (6) and is known as NP-hard. Therefore, in the BATS methodology, we relax this constraint by allowing $h_{i,p} \in \mathbb{R}$ to take any arbitrary value, and turn (10) into

$$\min_{H\in\mathbb{R}^{n\times k}} Tr(H^T LH) \text{ subject to } H^T H = I. \tag{11}$$

This approach allows us to employ clustering algorithms to solve the minimization problem. In the following sections, we detail our methods for solving (11) to cluster words (Section 3.3.1) and sentences (Section 3.3.2), respectively.

3.3.1 Topics via word clustering. To obtain the topics, we consider spectral clustering for the normalized matrix V' in (5). Since each row $v'_j \in \mathbb{R}^k, j = 1, ..., n$ of V is a k-dimensional representation of a word, the clustering optimization in (9) takes these n words as the nodes to be grouped into k sets $S_1, S_2, ..., S_k$ based on the similarities $g_{j,j'}$ between pairs of word representation vectors v_j and $v_{j'}$. This is equivalent to minimizing the pairwise deviations between representations of nodes within the sets:

$$S = \underset{\{S_1,...,S_k\}}{\arg\min} \sum_{p=1}^{k} \frac{\sum_{j,j'\in S_p} ||v'_j - v'_{j'}||^2}{2|S_p|}. \tag{12}$$

This optimization is equivalent to a KMeans clustering [27] of the vectors $v'_1, ..., v'_n$. The number of clusters is determined by the number of segments k, and each resulting word cluster S_p refers to a topic. To obtain a description of each topic in terms of its top words, we further rank the words in each cluster according to the standard term frequency-inverse document frequency (tf-idf) metric [39] applied to the awarded sentence-word matrix X^a in (1). The tf-idf assignment matrix F is obtained during the matrix decomposition process and it is the same size of X^a. To assign each word a single tf-idf score for sorting, we sum the tf-idf scores of each word over all sentences. The summed scores for every word are therefore used for sorting.

3.3.2 Segments via sentence clustering. We then turn to clustering the normalized matrix U' in (5) to obtain the segments. Compared with the topic clustering problem, this one will have more constraints since the clusters are related to the sentence orders and the cluster sizes can be largely uneven. As a result, the KMeans method is no longer applicable, and we resort instead to an agglomerative clustering method with connectivity constraints [13] to solve (9). In agglomerative clustering, nodes are grouped together sequentially according to their pairwise similarities: the process recursively merges two groups of nodes that yield the minimum between-cluster distance together into one group, with this merged cluster then being seen as a node in the following iterations.

Formally, recall that the sentence embeddings are the rows $u'_i \in \mathbb{R}^k, i = 1, ..., m$ of the matrix U'. We form the graph of sentences $G_S = (V, E_S)$, where $V = \{i|i = 1, 2, ..., m\}$ and $E_S = \{(i, j)|i, j = 1, 2, ..., m, i \neq j\}$, with the weight of the edge $(i, j) \in E_S$ being the cosine similarity between u'_i and u'_j. The ordering constraint should be such that only adjacent sentences can be clustered; we therefore initialize a connectivity graph $G_C^1 = (V, E_C^1)$ where for all pairs of nodes $i, j \in V$, $(i, j) \in E_C^1$ if and only if $j = i + 1$, i.e., each node connects to the next sentence. Letting S_i^r denote cluster i of

Algorithm 2 BATS topic modeling and text segmentation.

INPUT: Single text document, segment number k
PARAMETER: Awarding value λ, window size w, decaying rate d
OUTPUT: Topic words, text segments

```
procedure MainProcess(text, k, λ, w, d)
    Remove degree-one words from text.
    Compute sentence-word matrix X^o and POS-based matrix T.
    U′, V′, F ← MAT_DECOMP(X^o, T, λ, w, d, k). // Alg.1
    Cluster the rows of V′ with KMeans into k clusters. Sort the words in each cluster by F
    (tf-idf scores).
    Cluster the rows of U′ with agglomerative clustering into k clusters with a connectivity
    constraint.
    Topic words ← Sorted words in each cluster
    Text segments ← Sentence clusters
    return Topic words, Text segments
```

the sentences at the rth iteration, initialized as $S_i^1 = \{i\}$ for each i, the merging operation of our constrained agglomerative clustering is given by

$$
\begin{aligned}
(S_p^r, S_q^r) &= \underset{(i,j)\in E_C^r}{\arg\min}\ \overline{D}(S_i^r, S_j^r),\\
S_p^{r+1} &= S_p^r \cup S_q^r,\\
S_q^{r+1} &= \emptyset,\\
E_C^{r+1} &= E_C^r \setminus \{(p,q)\} \cup \{(p, a^r(q)\},
\end{aligned}
\tag{13}
$$

for $r = 1, ..., m - k$, where $\overline{D}(S_i^r, S_j^r)$ refers to the distance between the sets S_i^r and S_j^r, which is treated as the average distance between sentences in S_i^r and S_j^r according to their link weights in E_S, and $a^r(q) = v : (q, v) \in E_C^r$ is the single node that q points to in G_C^r. In each iteration, the two adjacent clusters of sentences that have minimum distance are merged together. The procedure ends after $r = m - k$ iterations, when there are k clusters i for which $S_i^k \neq \emptyset$; these are taken as the segments.

3.3.3 BATS methodology summary. The full BATS topic modeling and text segmentation methodology (including Algorithm 1) developed in this section is summarized in Algorithm 2. The inputs are the single text document of interest and k, the number of topics and segments to extract. The algorithm begins with denoising, which removes all degree-one words, and constructing the sentence-word matrix X^o and parts-of-speech matrix T. X^o and T are then inputted to the matrix decomposition procedure, detailed in Algorithm 1, which employs sentence bonding and graph Laplacian regularization to obtain the matrices U' and V', containing the encodings of the sentences and words, and the tf-idf matrix F. The rows of V' are then clustered into k clusters of words via KMeans, with the words in each cluster sorted by tf-idf score in F, forming the topics. Finally, the rows of U' are clustered into k clusters of sentences via constrained agglomerative clustering, forming the segments.

3.4 Time Complexity Analysis

We also perform a complexity analysis to investigate the efficiency of our algorithm. From Algorithm 1, note that there are three main procedures in BATS which have major impacts on the time complexity: matrix decomposition, KMeans clustering for words, and agglomerative clustering for sentences. The matrix decomposition process consists of multiple matrix multiplication and summation operations, of which matrix multiplication dominates with a complexity of $O(\max(m^3, n^3))$, where m and n are the number of sentences and words, respectively. However, in our application, the sentence-word matrix is sparse and therefore the matrix decomposition procedure can be done in a much less complex manner. Sparse matrices are usually stored in compressed sparse column (CSC) format, compressed sparse row (CSR) format, or triplet format. The complexity of matrix operations on these compressed formats depends mainly on the number of non-zero entries [7]. Formally, we can show that these non-zero entries in the matrix decomposition dominate the time complexity of BATS:

Lemma 1. *For a given document, let z be the number of non-zero entries in $X^a \in \mathbb{R}^{n\times m}$. For sufficiently large n, i.e., for sufficient diversity in the number of unique words comprising a document, the runtime of BATS can be approximated as $O(z^2/m)$.*

Proof. Since we are doing multiplication based on CSC or CSR format, from [7], the complexity of the matrix decomposition under sparse conditions is known to be

$$O\left(\max\left(\frac{z^2}{m}, \frac{z^2}{n}\right)\right), \tag{14}$$

where z is the number of non-zero entries. In the KMeans clustering procedure, all n word vectors are compared to k centroids to find the closest centroid, and this step iterates t_K times, leading to a time complexity of $O(nt_K k)$. In the constrained agglomerative procedure, the similarities between m sentence vectors are computed for clustering, and with t_A iterations, the time complexity is $O(t_A m \log m)$ with the efficient priority queue implementation [29]. The overall time complexity of our method is the sum of all these procedures, which leads to

$$O\left(\max\left(\frac{z^2}{m}, \frac{z^2}{n}\right) + nt_K k + t_A m \log m\right). \tag{15}$$

Noting that in most cases $n \gg m, t_K, t_A, k$, the term $\max(z^2/m,\ z^2/n)$ reduces to z^2/m, and the term $t_A m \log m$ can be ignored. Moreover, t_K, t_A, k are usually very small (less than 10) compared with m, and thus $nt_K k$ is dominated by n. Noting also that z grows with nm, z^2/m will dominate $nt_K k$. As a result, we can approximate the time complexity as $O(z^2/m)$ when n is sufficiently large. □

For the datasets considered in this paper (see Section 4), after preprocessing, we find that the average percentage of non-zero entries in a document's X^a matrix (see Table 1) are only 15.2% (for Introductions), 24.8% (for Textbook), 11.0% (for Lectures), and 23.5% (for Choi). These generate low expected computational complexities in Lemma 1 as the size of documents grow. The scalability of BATS will be verified experimentally in Section 4.4.

4 EXPERIMENTAL EVALUATION AND DISCUSSION

We turn now to evaluating our BATS methodology. After describing the datasets (Section 4.1), we consider performance against baselines on the topic modeling (Section 4.2) and text segmentation (Section 4.3) tasks. Finally, we consider the scalability of our method (Section 4.4). All experiments are conducted on a server with eight 4.2GHz Intel Core i7-7700k processors and 16 GB of memory.

Table 1. Basic statistics of the four datasets used for evaluation. The first three are used in topic modeling, while all four are used in text segmentation.

Dataset	Documents	Avg. sentences per doc	Avg. segments per doc	Avg. words before preproc.	Avg. words after preproc.	Avg distinct words before preproc.	Avg. distinct words after preproc.	Avg. sparsity before preproc.	Avg. sparsity after preproc.
Textbook	227	136	4	2590	3551	640	241	97.9%	75.2%
Lectures	55	392	8	4924	8002	686	342	99.0%	89.0%
Introductions	2135	195	5	4752	7022	1016	449	98.6%	84.8%
Choi	920	74	10 (const)	1673	1489	650	162	98.0%	76.5%

[a] The number of segments for each document in the Choi dataset is set to be constantly 10.

4.1 Description of Datasets

We consider documents from four datasets – Textbook, Lectures, Introductions, and Choi – obtained from different text applications. Basic statistics on these datasets are given in Table 1, including the number of documents, the average sentences per document, the average word counts per document, and the average sparsity per document (fraction of zero entries in the X^a matrix), before and after the preprocessing procedures of stopword removal, degree-one removal, and noun/verb awarding described in Section 3.1. More specifics on these datasets are as follows:

(i) Textbook dataset: This is drawn from the medical textbook in [47]. Each chapter is treated as a document, and each section as a segment. The numbers of segments per document and the numbers of sentences per segment have a high variance. Moreover, segments within a document tend to be similar in their constituent words, as they are different sections of the same chapter. As a result, this dataset helps us to test the algorithms on cases where documents have different segments discussing similar topics.

(ii) Lectures dataset: This dataset contains transcripts of conversational lectures on AI and physics topics.[3] As each lecture is divided into sections by the speaker, we treat lectures as documents and sections as segments. Each lecture script has 6-10 sections, and the topics of the sections are relatively independent. Compared with the other datasets, the sentences are more conversational, tending to be shorter and simpler. Therefore, this dataset helps us examine algorithm performance on lengthy conversational documents.

(iii) Introductions dataset: In this dataset, every document is an artificial combination of abstracts and introductions from academic articles in different fields.[4] We randomly choose 3-8 articles, extract the abstract and introduction as one sample, and combine multiple samples into one document. Each sample is treated as one segment in the text segmentation task. Compared with the other datasets, this will allow us to test on cases with large segment sizes, uneven segment lengths, and a diverse set of topics.

(iv) Choi dataset: This is a standard dataset [10] widely used to evaluate text segmentation approaches. The documents in the dataset are artificial combinations of the first ℓ sentences of the documents in the Brown corpus [19]. Each document has 10 segments, with few sentences per segment. Because the dataset lacks explicit topic distributions and contains mostly segments that are too short for topic modeling, we use it only for evaluating text segmentation.

4.2 Topic Modeling

We first consider the performance of BATS in topic modeling on the Textbook, Lectures, and Introductions datasets compared with five standard baselines.

4.2.1 Topic modeling baselines. We compared BATS against five state-of-the-art baselines for topic modeling:

[3]These are from https://github.com/jacobeisenstein/bayes-seg/tree/master/data/lectures.

[4]These are taken from the sentence classification datasets at https://archive.ics.uci.edu/ml/datasets/Sentence+Classification.

	Topic 1	Topic 2
Human Summary	*"Effects of protein epsin on a membrance with clathrin-coat in eukaryotic cells."*	*"Investigate the conditions of existence of the energy function."*
BATS	membrane protein coat clathrin result vesicle suggest cell domain interaction	function energy study potential algorithm symbol demonstrate rule framework adaption
LSA	clathrin epsin coat **membrane** **energy** **protein** vesicle **parameter** bind **symbol**	citation model number **parameter** function **protein** **symbol** **energy** **membrane** use
LDA	make membrane parameter clathrin study behavior describe work problem experiment	function show analysis algorithm morphology heuristic space apt inference example

Fig. 3. Example of topics extracted from an arbitrary document in the Introductions dataset. Words in color red are those consistent with a human-generated summary, and duplicated words are boldfaced. Our results produce the best descriptions as well as the least overlaps.

(i) Latent Dirichlet Allocation (LDA) [3]: LDA is a probabilistic topic model which uses two independent Dirichlet priors for the document-topic and word-topic distributions. It trains a model to best estimate the Bayesian probabilities $P(word|topic)$ and $P(topic|document)$. We use the `sklearn` implementation in Python with the default parameters.
(ii) Hierarchical Dirichlet Process (HDP) [44]: HDP is a mixed-membership model which extends LDA to an unknown number of topics by building a hierarchy. Specifically, it builds a two-level hierarchical Dirichlet process at the document-level and the word-level to perform parameter inference. We use the `gensim` implementation in Python with the default parameters.
(iii) Latent Semantic Analysis (LSA) [14]: LSA decomposes a document-word matrix, based on TF-IDF scores, into a document-topic matrix and a topic-word matrix; the decomposition is performed through a truncated SVD technique. We use the `gensim` implementation in Python.
(iv) Probabilistic Latent Semantic Analysis (pLSA) [24]: pLSA is developed from LSA, using a probabilistic method instead of SVD to find the latent topics via generative modeling of the observed document-word matrix. We implement pLSA de-novo in Python, using 30 as the max number of iterations, 10.0 as the breaking threshold, and k as the number of topics.
(v) Non-negative Matrix Factorization (NMF) [35]: NMF is a linear-algebraic model which factorizes a high-dimensional matrix into two lower-dimensional ones. In this case, NMF decomposes the document-word matrix (based on TF-IDF scores) into a topic matrix and a coefficient matrix for the topics. We use the `sklearn` implementation in Python with the default parameters.

Since our focus is on single document topic modeling, we evaluate the models on each document separately. Given that the baselines usually learn across multiple documents, to provide a fair comparison, we treat the sentences within each document as the "documents" for the baselines, i.e., we feed them the preprocessed sentence-word matrices. For each document, the number of topics assumed by each baseline is taken to be the number of segments. The performance of each baseline is averaged over several trials.

4.2.2 Evaluation metrics. We employ two popular coherence metrics to assess extracted topic: CV [42] and UMass [32]. Higher values of these metrics have been associated with better performance in terms of interpretability and consistency of topics with human evaluation [42]. Since these metrics treat topics separately, in order to evaluate the diversity between topics, we also include two similarity measures: Jaccard (Jacc) Index and Sørensen-Dice (Dice) Index [22]. They measure overlaps in words between the topics, with lower values (i.e., less overlap) being better. More specifically:

T1	**hypothesis**	**patient**	**problem**	**examination**	**symptom**
	laboratory	**sign**	**history**	**physical**	**disease**
T2	**patient**	**problem**	**examination**	**symptom**	**hypothesis**
	sign	**physical**	pain	mark	palpate
T3	**symptom**	**hypothesis**	**sign**	**patient**	**disease**
	laboratory	analysis	**history**	pain	problem
T4	palpate	examine	**examination**	ascites	**patient**
	sign	system	**problem**	skin	**disease**
T5	**laboratory**	**hypothesis**	**examination**	**physical**	**history**
	patient	data	palpate	process	physician

Fig. 4. Example of topics extracted from one document (known to have five topics) in the Textbook dataset by LSA. Duplicated words are denoted in boldface. There is high overlap, motivating the need to consider topic diversity in addition to coherence.

(i) Topic coherence measures: CV is an extrinsic metric which uses an external corpus (i.e., a different corpus from the dataset under consideration) to compute empirical probabilities of each topic word. It then checks word co-occurrences within a Boolean sliding window, computes the normalized pointwise mutual information, and averages the results. By contrast, UMass is an intrinsic evaluation metric which takes the sequence of words into consideration by computing the conditional log-probability of each pair of words; the pairwise scores are not symmetric, and therefore the order of the words matters. In our single-document evaluation, we consider the external corpus for CV to be the dataset from which that document originates, and the internal corpus for UMass to be the document itself.

(ii) Similarity score measures: With T_i and T_j as the sets of words comprising topics i and j, the Jacc $Jacc(T_i, T_j)$ and Dice $Dice(T_i, T_j)$ similarity scores are computed as:

$$Jacc(T_i, T_j) = \frac{|T_i \cap T_j|}{|T_i \cup T_j|}, \tag{16}$$

$$Dice(T_i, T_j) = \frac{2|T_i \cap T_j|}{|T_i| + |T_j|}. \tag{17}$$

To see the importance of considering both types of metrics, consider the example in Figure 3, which shows topics extracted from an arbitrary document. Those extracted by the LSA baseline tend to have many duplicated words (50% in the example) as compared with results from LDA and BATS, even though it has roughly the same number of words that are consistent with a human-generated summary as our method. Further, since the overall scores for each document are averaged across topics, poor results in terms of one metric on any given topic can be outweighed by high performance on other topics. Since the overall topic coherence scores for each document are averaged across topics, similar topics with duplicate words and high coherence scores will achieve a high average score. Figure 4 shows another example of this for LSA: though this method achieves high topic coherence, the topics are highly overlapped, motivating the need to take diversity into consideration.

As a result, we also define composite metrics for evaluation which penalize the coherence scores on pairs of topics according to the similarity scores. Specifically, using $\mathtt{CV}_i$ and $\mathtt{UMass}_i$ as the coherence scores for topic i and $\mathtt{sim}_{i,j}$ as the similarity score between topics i and j according to

Table 2. Performance of each algorithm on the Introductions, Textbook and Lectures datasets in terms of topic coherence, similarity, and composite metrics. The means and standard deviations across documents are shown. Our algorithm has the highest performance on most of the metrics, indicating it achieves the best balance between topic coherence and diversity.

Textbook Dataset								
	Jacc	Dice	CV	$\boldsymbol{CV^{Jacc}}$	$\boldsymbol{CV^{Dice}}$	UMass	$\boldsymbol{UMass^{Jacc}}$	$\boldsymbol{UMass^{Dice}}$
LDA	**0.00 ± 0.00**	**0.00 ± 0.00**	0.45 ± 0.07	0.45 ± 0.07	0.45 ± 0.07	−14.74 ± 2.78	−14.74 ± 2.78	−14.74 ± 2.78
HDP	0.01 ± 0.01	0.02 ± 0.02	0.33 ± 0.05	0.32 ± 0.05	0.32 ± 0.05	−22.30 ± 0.61	−22.52 ± 0.68	−22.72 ± 0.80
LSA	0.28 ± 0.10	0.42 ± 0.13	**0.57 ± 0.08**	0.41 ± 0.08	0.33 ± 0.09	**−8.11 ± 2.21**	**−10.38 ± 2.96**	−11.49 ± 3.30
pLSA	0.10 ± 0.09	0.14 ± 0.12	0.35 ± 0.10	0.33 ± 0.09	0.31 ± 0.09	−14.65 ± 2.82	−16.06 ± 3.34	−16.78 ± 3.77
NMF	0.21 ± 0.10	0.31 ± 0.13	0.47 ± 0.09	0.37 ± 0.07	0.32 ± 0.07	−13.41 ± 2.82	−15.94 ± 3.99	−17.22 ± 4.28
BATS	**0.00 ± 0.00**	**0.00 ± 0.00**	0.53 ± 0.07	**0.53 ± 0.07**	**0.53 ± 0.07**	−11.37 ± 2.45	−11.37 ± 2.45	**−11.37 ± 2.45**
Lectures Dataset								
	Jacc	Dice	CV	$\boldsymbol{CV^{Jacc}}$	$\boldsymbol{CV^{Dice}}$	UMass	$\boldsymbol{UMass^{Jacc}}$	$\boldsymbol{UMass^{Dice}}$
LDA	**0.00 ± 0.00**	**0.00 ± 0.00**	0.39 ± 0.05	0.39 ± 0.05	0.39 ± 0.05	−14.60 ± 2.52	−14.60 ± 2.52	−14.60 ± 2.52
HDP	0.01 ± 0.01	0.01 ± 0.01	0.38 ± 0.04	0.38 ± 0.04	0.38 ± 0.04	−21.55 ± 0.34	−21.72 ± 0.37	−21.86 ± 0.42
LSA	0.27 ± 0.07	0.41 ± 0.09	**0.53 ± 0.08**	0.38 ± 0.06	0.31 ± 0.05	**−7.13 ± 1.76**	**−8.97 ± 2.12**	**−9.92 ± 2.34**
pLSA	0.04 ± 0.03	0.07 ± 0.04	0.36 ± 0.04	0.35 ± 0.04	0.34 ± 0.04	−19.44 ± 0.78	−20.27 ± 1.04	−20.901.31
NMF	0.26 ± 0.08	0.39 ± 0.10	0.47 ± 0.05	0.35 ± 0.04	0.29 ± 0.05	−9.07 ± 2.30	−11.39 ± 2.84	−12.51 ± 3.10
BATS	**0.00 ± 0.00**	**0.00 ± 0.00**	0.48 ± 0.04	**0.48 ± 0.04**	**0.48 ± 0.04**	−10.80 ± 1.75	−10.80 ± 1.75	−10.80 ± 1.75
Introductions Dataset								
	Jacc	Dice	CV	$\boldsymbol{CV^{Jacc}}$	$\boldsymbol{CV^{Dice}}$	UMass	$\boldsymbol{UMass^{Jacc}}$	$\boldsymbol{UMass^{Dice}}$
LDA	**0.00 ± 0.00**	**0.00 ± 0.00**	0.30 ± 0.04	0.30 ± 0.04	0.30 ± 0.04	−15.38 ± 1.72	−15.38 ± 1.72	−15.38 ± 1.72
HDP	0.01 ± 0.01	0.01 ± 0.02	0.33 ± 0.04	0.33 ± 0.04	0.33 ± 0.04	−21.78 ± 1.61	−21.92 ± 1.61	−22.04 ± 1.62
LSA	0.21 ± 0.09	0.31 ± 0.12	0.43 ± 0.07	0.34 ± 0.07	0.30 ± 0.06	**−8.11 ± 2.03**	−9.88 ± 2.77	−10.77 ± 3.11
pLSA	0.04 ± 0.05	0.06 ± 0.07	0.35 ± 0.09	0.35 ± 0.09	0.35 ± 0.09	−16.15 ± 2.29	16.76 ± 2.39	−17.16 ± 2.56
NMF	0.21 ± 0.08	0.32 ± 0.11	0.31 ± 0.05	0.25 ± 0.04	0.21 ± 0.04	−14.18 ± 2.32	−17.05 ± 2.61	−18.50 ± 2.85
BATS	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.49 ± 0.05**	**0.49 ± 0.05**	**0.49 ± 0.05**	−8.18 ± 1.68	**−8.18 ± 1.68**	**−8.18 ± 1.68**

[a] The titles of composite metrics are highlighted in boldface.
[b] The best results on each metric are highlighted in boldface.

Jacc or Dice, we compute the following:

$$CV^{sim} = \frac{\sum_{i=1}^{k}\sum_{j=1}^{k}(\mathtt{CV}_i + \mathtt{CV}_j)(1 - \mathtt{sim}_{i,j})/2}{k^2}, \tag{18}$$

$$UMass^{sim} = \frac{\sum_{i=1}^{k}\sum_{j=1}^{k}(\mathtt{UMass}_i + \mathtt{UMass}_j)(1 + \mathtt{sim}_{i,j})/2}{k^2}, \tag{19}$$

where k refers to the total number of topics. Since CV scores are positive and UMass scores are negative, penalties are set as $1 - \mathtt{sim}_{i,j}$ and $1 + \mathtt{sim}_{i,j}$, respectively.

4.2.3 Results and discussion. The results obtained by each algorithm on the three datasets are given in Table 2. We present the mean and standard deviations on topic diversity, topic coherence, and the four cases of joint metrics. The first two columns, Jacc and Dice, indicate the diversities of the topics (smaller being better). The following columns then give the topic coherence scores, CV and UMass (larger being better), followed by their combinations with the similarity measures (e.g., CV^{Dice} is CV with Dice used for $\mathtt{sim}_{ij}$ in (18)).

Overall, we see that compared with the baselines, *our method BATS obtains competitive topic coherence scores, the lowest similarity scores, and the best composite scores in most cases.* For the Introductions dataset, our method maintains higher performance than all baselines in all metrics except UMass. On the Textbook and Lectures datasets, our method still obtains the highest performance in the most cases. Across the datasets, the percent improvements over the strongest baselines in

Table 3. Text segmentation evaluation metrics obtained on each of the four datasets. The average and standard deviation of each metric across documents is shown. Our method outperforms the baselines in all cases, except for C99 on the Choi dataset.

	Textbook Dataset		Lectures Dataset		Introductions Dataset		Choi Dataset	
	P_k	WD	P_k	WD	P_k	WD	P_k	WD
TextTiling	0.45 ± 0.177	0.47 ± 0.169	0.43 ± 0.099	0.48 ± 0.088	0.29 ± 0.158	0.33 ± 0.170	0.33 ± 0.076	0.34 ± 0.075
C99	0.55 ± 0.142	0.79 ± 0.206	0.51 ± 0.075	0.85 ± 0.164	0.20 ± 0.120	0.29 ± 0.184	**0.14 ± 0.077**	**0.15 ± 0.081**
TopSeg	0.58 ± 0.128	0.71 ± 0.091	0.60 ± 0.088	0.75 ± 0.084	0.59 ± 0.090	0.72 ± 0.085	0.41 ± 0.056	0.44 ± 0.062
LDA_MDP	0.52 ± 0.161	0.60 ± 0.121	0.53 ± 0.117	0.63 ± 0.118	0.51 ± 0.142	0.59 ± 0.136	0.49 ± 0.079	0.050 ± 0.088
TopicTiling	0.50 ± 0.157	0.56 ± 0.140	0.51 ± 0.110	0.56 ± 0.100	0.50 ± 0.138	0.57 ± 0.122	0.45 ± 0.079	0.47 ± 0.082
BATS	**0.42 ± 0.167**	**0.44 ± 0.165**	**0.41 ± 0.161**	**0.46 ± 0.118**	**0.16 ± 0.134**	**0.18 ± 0.147**	0.22 ± 0.103	0.23 ± 0.109

[a] The scores in boldface are the best performing ones for each dataset and metric.

the composite CV metrics are between 18% and 40%. The baseline which tends to outperform our algorithm in terms of topic coherence, LSA, also performs the worst in terms of topic diversity. To interpret this diversity performance, we note that a Jacc score of 0.25 and a Dice score of 0.4 correspond roughly to $|T_i \cap T_j| \propto 0.4$ in (16),(17), i.e., a 40% duplication between topics. Thus, LSA (as well as NMF) usually has up to 40% average overlap in topic words, leading to confusing topics, while our method yields no noticeable overlap. On the other hand, the baseline which matches our algorithm in topic diversity, LDA, is among the lowest performing in terms of coherence, which is also reflected in the composite metrics. We can thus conclude that, among the algorithms tested, *our algorithm finds the best balance between topic coherence and diversity* for single document topic modeling; its consistent performance across the datasets also shows that it is robust to variations in dataset properties like segment and document length.

We also observe an interesting pattern in the baselines: the spectral methods – LSA and NMF – perform high in coherence but low in similarity, while the generative models – LDA and pLSA – have the opposite trends. While spectral approaches can extract topics that are interpretable when taken individually, there is high similarity between them because they are based on matrix decomposition and do not consider diversity. Generative models can extract diversified topics, but when they are operating on single documents with few word co-occurrences, the resulting topics will not be as coherent. These observations are consistent with the comparison between LSA, LDA, and BATS in Figure 3.

4.3 Text Segmentation

Next, we consider the performance of BATS in text segmentation on the four datasets compared with five standard baselines.

4.3.1 Text segmentation baselines. We compared BATS against five state-of-the-art baselines for the text segmentation task:
(i) TextTiling [23]: TextTiling divides the text into pseudosentences, assigns similarity scores at the gaps, detects peak differences in the scores, and marks the peaks as boundaries. The boundaries are normalized to the closest sentence breaks. We use the implementation from the `nltk` package in Python.
(ii) C99 [10]: C99 is another popular text segmentation algorithm that inserts boundaries based on average inter-sentence similarities. More specifically, a ranking transformation is performed, pairwise cosine distances between sentences are computed based on the ranking, and boundaries are determined based on these similarities. We implement C99 de-novo in Python.
(iii) TopSeg [4]: TopSeg is a text segmentation method which make use of the pLSA topic modeling technique. The model requires pretraining the topic model on the targeting dataset, and the text

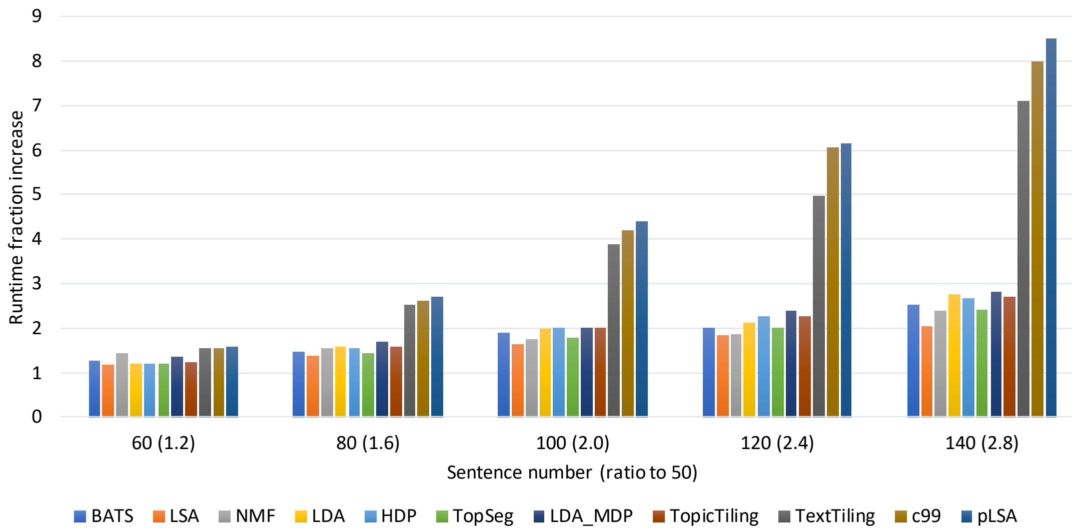


Fig. 5. Factor increase in runtime when varying the number of sentences in each segment for the Choi dataset. The time increase is relative to the case of 50 sentences, and each bar is an average over 10 runs. Our method scales well compared with the baselines.

segmentation process is then dependent on the trained topic model. We implement the TopSeg model de-novo in Python.

(iv) Modified DP Algorithm with LDA (LDA_MDP) [33]: LDA_MDP performs text segmentation based on another topic model, LDA, with the segmentation being implemented with dynamic processing (DP) techniques. The method has also been tested using an alternate topic model, multinomial mixture, but LDA has has been found to obtain better performance. We implement the LDA_MDP model de-novo in Python.

(v) TopicTiling [41]: TopicTiling is based on TextTiling, with additionally integrated topic information from the LDA topic model for text segmentation. We implement TopicTiling de-novo in Python, using a window size of 2 and 500 iterations.

To say consistent across the algorithms, note that for the topic-based text segmentation methods – TopSeg, LDA_MDP, and TopicTiling – we train the topic model with the single document that it is segmenting.

4.3.2 Evaluation metrics. We consider two standard text segmentation metrics, P_k [2] and WindowDiff (WD) [37]. Lower values indicate better performance. Each of these metrics compares the ground truth (i.e., reference) segmentation `ref` to the estimated (i.e., hypothesized) segmentation `hyp`. The P_k metric calculates the number of disagreements in the positions of segment boundaries between `hyp` and `ref`; in doing so, it ignores the exact number of boundaries to be detected, and weights false positives more heavily [37]. WD, on the other hand, slides a fixed-sized window across the document, calculates the number of boundaries within that window, and records an error if `ref` and `hyp` disagree on the number.

Formally, let $(x_1, x_2, ..., x_N)$ be the sequence of N words comprising a document, where each $x_i \in \mathcal{W}$, the set of document words. With $\delta_z(i, j)$ as the binary indicator of whether words x_i and x_j are in the same segment under segmentation z, and $b_z(i, j)$ as the number of segment boundaries between x_i and x_j under z, the metrics are calculated as

$$P_k = \frac{1}{N-\ell} \sum_{i=1}^{N-\ell} \mathbb{1}\{|\delta_{\mathsf{hyp}}(i, i+\ell) - \delta_{\mathsf{ref}}(i, i+\ell)| > 0\}, \tag{20}$$

$$\mathrm{WD} = \frac{1}{N-\ell} \sum_{i=1}^{N-\ell} \mathbb{1}\{|b_{\mathsf{hyp}}(i, i+\ell) - b_{\mathsf{ref}}(i, i+\ell)| > 0\}, \tag{21}$$

where the window size ℓ is set to one less than half the average segment length, and $\mathbb{1}$ is the indicator function.

4.3.3 Results and discussion. The results obtained by each algorithm on each of the four datasets are given in Table 3. The mean and standard deviation across documents is shown in each case.

Overall, we see that *our method BATS consistently outperforms all of the baselines in terms of text segmentation on the first three datasets*. The highest performing baseline changes depending on the dataset, with TextTiling being most competitive on Textbooks and Lectures, and C99 being most competitive on Introductions and Choi. The three topic-based text segmentation methods (TopSeg, LDA_MDP, and TopicTiling) actually perform considerably worse than these other baselines, possibly due to single documents containing insufficient data for training their topic models (recall in particular that LDA had poor topic diversity performance in Table 3). Combined with the results in Sec. 4.2, we conclude that *our method is capable of identifying accurate segment boundaries and topic words for a single document simultaneously*.

On the Choi dataset, BATS outperforms each of the baselines except C99. C99 is designed specifically with datasets such as Choi in mind, where documents are artificially built with identical numbers of short segments and sparse content in each segment [10]. Specifically, as shown in Table 1, the average sentences per segment and per document in Choi are significantly smaller than the other three datasets. This is due to the way it is constructed – with each document as combinations of first ℓ sentences from documents in another corpus – making it less realistic than the other datasets.

4.4 Scalability Analysis

Finally, we evaluate the effect of the number of sentences and segments on the runtime of our method compared with the baselines. Figure 5 shows the increase in runtime from varying the number of sentences in each segment for the Choi dataset, relative to the case of 50 sentences (we choose this dataset because all documents are constructed with a constant number of segments). We can see that *the growth in runtime of our methodology BATS is comparable to the most scalable baselines*, with the rate of increase in runtime less than the corresponding increase in sentences. Additionally, BATS is the only methodology performing both topic modeling and text segmentation. Out of the baselines in Figure 5, TextTiling, c99, and pLSA have considerably higher increases in runtime, with pLSA performing the worst. The substantial difference between LSA, the most scalable, and pLSA is consistent with spectral approaches being known to scale better than generative algorithms that require multiple iterations [52].

Figure 6 shows the impact on runtime from varying the number of segments per document for the Lectures dataset (recall from Table 1 this dataset has the longest documents available). Here we have excluded pLSA, as its runtime is significantly longer, and also the text segmentation baselines, as their runtimes are not dependent on the number of segments. NMF is by far impacted the most, followed by LSA, while HDP and LDA exhibit the best scalability. Our method remains impacted under 10% for a 5-fold increase in segments, again implying that *our method supports changes in the*

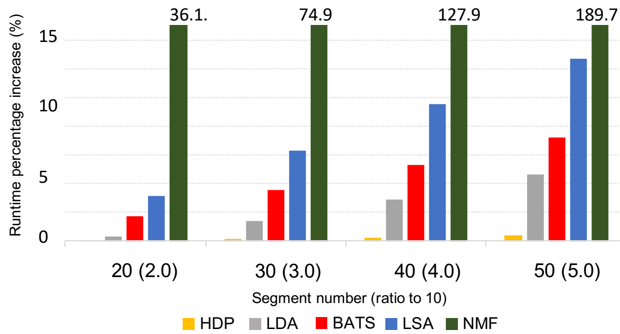


Fig. 6. Increase in runtime when varying the number of segments per document in the Lectures dataset. The baseline is 10 segments, and each bar is over 10 runs. Our method again scales well compared with the baselines.

size of input efficiently. Taken together, Figures 5 and 6 validate our theoretical analysis in Section 3.4 which concluded that BATS has low computational complexity.

5 CONCLUSION AND FUTURE WORK

In this work, we developed an unsupervised, computationally efficient, statistically sound methodology called BATS that simultaneously extracts the topics and segments the text from one single document. BATS first leverages word-order information together with optimization tricks such as parts-of-speech (POS) tagging to refine a document's sentence-word matrix. It then obtains a singular value decomposition from a regularized form of the graph Laplacian, with the singular vectors yielding low dimensional embeddings of words and sentences. Finally, BATS employs clustering algorithms to extract topics and text segments from the left and right singular vectors. Through evaluations against five topic modeling baselines on three datasets, and against five text segmentation baselines on four datasets, we confirmed that our algorithm achieves the best overall performance on standard metrics in both topic extraction and segmentation tasks. For topic extraction, this was especially true when considering the dual objectives of coherence maximization and similarity minimization across topics. Our experimental results also showed that BATS scales well with the size of the input data, and that it is robust to changes in dataset characteristics such as document lengths and segment numbers, which confirmed our preceding analysis on computational complexity.

We identify several potential avenues of future work. A more elaborate POS awarding scheme in the sentence-word matrix construction phase may improve topic coherence further. Since BATS provides both topic and text segment information, the application of our methodology to text summarization can also be considered, e.g., in identifying the most important segments according to the number of corresponding topic words, or in generating a set of keywords from topic words for the entire document. Text summarization on single documents would also be useful for the motivating examples given in Section 1, e.g., in rapidly summarizing emerging news events for users.

REFERENCES

[1] Arash A Amini, Aiyou Chen, Peter J Bickel, Elizaveta Levina, et al. 2013. Pseudo-likelihood methods for community detection in large sparse networks. *The Annals of Statistics* 41, 4 (2013), 2097–2122.
[2] Doug Beeferman, Adam Berger, and John Lafferty. 1999. Statistical models for text segmentation. *Machine learning* 34, 1-3 (1999), 177–210.

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

[4] Thorsten Brants, Francine Chen, and Ioannis Tsochantaridis. 2002. Topic-based document segmentation with probabilistic latent semantic analysis. In *Proceedings of the eleventh international conference on Information and knowledge management*. ACM, 211–218.

[5] Christopher G Brinton, Swapna Buccapatnam, Liang Zheng, Da Cao, Andrew S Lan, Felix MF Wong, Sangtae Ha, Mung Chiang, and H Vincent Poor. 2018. On the efficiency of online social learning networks. *IEEE/ACM Transactions on Networking* 26, 5 (2018), 2076–2089.

[6] Christopher G Brinton and Mung Chiang. 2016. *The power of networks: Six principles that connect our lives.* Princeton University Press.

[7] Aydin Buluç, Jeremy T Fineman, Matteo Frigo, John R Gilbert, and Charles E Leiserson. 2009. Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks. In *Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*. 233–244.

[8] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in neural information processing systems*. 288–296.

[9] Kamalika Chaudhuri, Fan Chung, and Alexander Tsiatas. 2012. Spectral clustering of graphs with general degrees in the extended planted partition model. In *Conference on Learning Theory*. 35–1.

[10] Freddy YY Choi. 2000. Advances in domain independent linear text segmentation. *arXiv preprint cs/0003083* (2000).

[11] Wanqiu Cui, Junping Du, Dawei Wang, Xunpu Yuan, Feifei Kou, Liyan Zhou, and Nan Zhou. 2019. Short Text Analysis Based on Dual Semantic Extension and Deep Hashing in Microblog. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 4 (2019), 1–24.

[12] Ali Daud, Juanzi Li, Lizhu Zhou, and Faqir Muhammad. 2010. Knowledge discovery through directed probabilistic topic models: a survey. *Frontiers of computer science in China* 4, 2 (2010), 280–301.

[13] Ian Davidson and SS Ravi. 2005. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 59–70.

[14] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41, 6 (1990), 391–407.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[16] Inderjit S Dhillon. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In *ACM SIGKDD*. ACM, 269–274.

[17] Inderjit S Dhillon, Subramanyam Mallela, and Dharmendra S Modha. 2003. Information-theoretic co-clustering. In *ACM SIGKDD*. ACM, 89–98.

[18] Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14, 2 (1990), 179–211.

[19] W. N. Francis and H. Kucera. 1979. *Brown Corpus Manual.* Technical Report. Department of Linguistics, Brown University, Providence, Rhode Island, US. http://icame.uib.no/brown/bcm.html

[20] Yang Gao, Yuefeng Li, Raymond YK Lau, Yue Xu, and Md Abul Bashar. 2017. Finding semantically valid and relevant topics by association-based topic selection model. *ACM Transactions on Intelligent Systems and Technology (TIST)* 9, 1 (2017), 1–22.

[21] Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. *COLING* (2012), 959–976.

[22] Wael H Gomaa and Aly A Fahmy. 2013. A survey of text similarity approaches. *International Journal of Computer Applications* 68, 13 (2013), 13–18.

[23] Marti A Hearst. 1997. TextTiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics* 23, 1 (1997), 33–64.

[24] Thomas Hofmann. 2017. Probabilistic latent semantic indexing. In *ACM SIGIR*, Vol. 51. ACM, 211–218.

[25] Yuval Kluger, Ronen Basri, Joseph T Chang, and Mark Gerstein. 2003. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome research* 13, 4 (2003), 703–716.

[26] Andrew S Lan, Jonathan C Spencer, Ziqi Chen, Christopher G Brinton, and Mung Chiang. 2018. Personalized thread recommendation for MOOC discussion forums. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 725–740.

[27] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* 28, 2 (1982), 129–137.

[28] Okumura Manabu and Honda Takeo. 1994. Word sense disambiguation and text segmentation based on lexical cohesion. In *The 15th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, 755–761.

[29] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval.* Cambridge university press.

[30] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*. 6294–6305.

[31] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*. 3111–3119.

[32] David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 262–272.

[33] Hemant Misra, François Yvon, Olivier Cappé, and Joemon Jose. 2011. Text segmentation: A topic modeling perspective. *Information Processing & Management* 47, 4 (2011), 528–544.

[34] Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015. Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics* 3 (2015), 299–313.

[35] V Paul Pauca, Farial Shahnaz, Michael W Berry, and Robert J Plemmons. 2004. Text mining using non-negative matrix factorizations. In *SIAM International Conference on Data Mining*. SIAM, 452–456.

[36] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[37] Lev Pevzner and Marti A Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics* 28, 1 (2002), 19–36.

[38] Tai Qin and Karl Rohe. 2013. Regularized spectral clustering under the degree-corrected stochastic blockmodel. In *Advances in Neural Information Processing Systems*. 3120–3128.

[39] Anand Rajaraman and Jeffrey David Ullman. 2011. Mining of Massive Datasets: Data Mining (Ch01). *Min. Massive Datasets* 18 (2011), 114–142.

[40] Philip Stuart Resnik. 1993. Selection and information: a class-based approach to lexical relationships. *IRCS Technical Reports Series* (1993), 200.

[41] Martin Riedl and Chris Biemann. 2012. TopicTiling: a text segmentation algorithm based on LDA. In *ACL Student Research Workshop*. Association for Computational Linguistics, 37–42.

[42] Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the space of topic coherence measures. In *ACM international conference on Web search and data mining*. ACM, 399–408.

[43] Karl Rohe, Tai Qin, and Bin Yu. 2012. Co-clustering for directed graphs: the Stochastic co-Blockmodel and spectral algorithm Di-Sim. *arXiv:1204.2296* (2012).

[44] Yee W Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2005. Sharing clusters among related groups: Hierarchical Dirichlet processes. In *Advances in neural information processing systems*. 1385–1392.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[46] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17, 4 (2007), 395–416.

[47] Henry Kenneth Walker, Wilbur Dallas Hall, and John Willis Hurst. 1990. *The Oral Cavity and Associated Structures–Clinical Methods: The History, Physical, and Laboratory Examinations.* Butterworths.

[48] Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A biterm topic model for short texts. In *International conference on World Wide Web*. ACM, 1445–1456.

[49] Jianhua Yin and Jianyong Wang. 2014. A dirichlet multinomial mixture model-based approach for short text clustering. In *ACM SIGKDD*. ACM, 233–242.

[50] Qi Zhang, Yang Wang, Yeyun Gong, and Xuan-Jing Huang. 2016. Keyphrase extraction using deep recurrent neural networks on twitter. In *Proceedings of the 2016 conference on empirical methods in natural language processing*. 836–845.

[51] He Zhao, Lan Du, Wray Buntine, and Gang Liu. 2017. MetaLDA: a topic model that efficiently incorporates meta information. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 635–644.

[52] Shi Zhong and Joydeep Ghosh. 2005. Generative model-based document clustering: a comparative study. *Knowledge and Information Systems* 8, 3 (2005), 374–384.