Scalable Topic Models on Live Streamed Speech Transcription

James Timotiwu
University of Illinois at Urbana-Champaign

Andrew Ma
University of Illinois at Urbana-Champaign

Alpri Else
University of Illinois at Urbana-Champaign

Abstract

In this paper, we explore how topic detection can be done in real-time on transcription text corpora. Concretely, this paper aims to answer the following questions: how can topics in transcriptions evolve over time as live streamed content changes? Can topic detection across multiple streams be done at scale without compromising latency when thousands or millions of conversations could be occurring concurrently? Topic modeling in real-time is difficult in streaming scenarios because data is unbounded. Furthermore, online learning systems suffer from poor resource utilization compared to batch approaches. This can be wasteful due to the heterogeneity of streamed data, unreliable connectivity, and the non-deterministic delivery of data. We introduce a system that enables real-time distributed topic-modeling and document-topic distribution inference on streamed voice transcription data. The system implements a novel real-time text shift detection module to allow the scheduler to make informed decisions on data partitioning. The new partitioning strategy should also help improve the locality of sparse parameters in the model, which would make efficient use of the cache.

1 Introduction

With the growing amount of large unstructured text data, understanding these collections has been a difficult issue. Understanding these documents and categorizing them using unsupervised learning is a way for many to scalably bring structure to these datasets. Topic modeling has become a widely popular statistical model that intends to address this problem. Topic models naturally group these texts into latent "topics". We use Latent Dirichlet allocation (LDA), a generative probabilistic model, to discover topics within a document.

Topic models are used in a wide variety of applications including social event analysis [17], image processing [1], and software engineering [10]. Content-based Recommendation Systems are used prevantly in many social media apps, and topic models are commonly used to group similar pieces of content together. Topic modeling used in concert with collaborative filtering techniques in hybrid Recommender Systems have also shown potential in addressing common issues like the cold-start problem [20,21]. With the rise of social media apps with live content, namely, Twitch and Clubhouse, recommender systems in these novel environments need to adapt to changing topics while content is streamed into the system live.

Recent research of topic modeling has looked into the question of scaling such models. These advancements in topic modeling are looked at through the lens of distributed systems [6,16,24]. Optimized parallelization strategies are researched in order to reach massive data and model scales.

Standard topic modeling algorithms, such as LDA, do not address evolving environments, but some variants have been introduced that address some special cases. Many of these look into the

time evolution of topics within these text documents. Dynamic Topic Models (DTM) looks into situations where topic-word distribution varies over time [4]. Another variation, SeqLDA, considers the sequential structure of text documents and the relation between previous and subsequent segments [9]. While these techniques consider concept drifts and look into situations where topic models evolve over time, these temporal variants don't consider the documents changing over time themselves.

This paper looks into an industry applicable area of topic modeling in dynamic live streamed environments where a document's topic distribution needs to be continually updates as more content is streamed into the system. This is relevant because with the growing number of data used in many companies, it is important to prevent unnecessary retraining of machine learning models, such as topic models. Existing work has not addressed changing document-topic distributions in contemporary live streamed environments. Latency is an issue where recommendations for live streamed content need to be updated as live stream occurs. Parallelism is another commanding issue where there needs to be a distributed learning/parameter server system design for any solution to work across thousands of simultaneous livestreams. Other issues unaddressed in research papers are online and streaming capabilities, drift and segmentation of topics within corpus, and real-time inference.

Overall, this paper intends to make the following contributions:

- Identifies live-streamed content as a novel environment for topic modeling that has direct applications to industry recommender systems.
- Proposes and studies different approaches to assigning topic distributions to documents being streamed into our system.
- A system model for dealing with novel distributed systems issues in this environment.

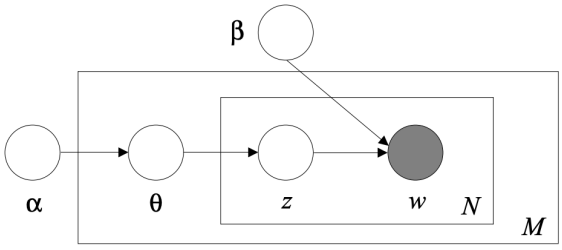

Figure 1: Graphical model representation of LDA taken from [5] where w_n is the nth observed word in document w drawn from $p(w_n|z_n,\beta)$.

2 Background

2.1 Latent Dirichlet Allocation Basics

Latent Dirichlet Allocation (LDA) [5] is a hierarchical Bayesian probabilistic model for generatively assigning latent topic proportions to documents. In LDA, documents are represented as random mixtures over latent topics where each topic is a multinomial distribution over the vocabulary. More formally, each document $w = (w_1, w_2, ..., w_N)$ is associated with topic proportions θ and z_n, the topic assignment for a given word w is drawn from $Multinomial(\theta)$. Since we are interested in each document's topic proportions θ and each word's topic assignment z, we need to solve for the posterior distribution of the hidden variables given a document [5]:

$$p(\theta, z|w, \alpha, \beta) = \frac{p(\theta, z, w|\alpha, \beta)}{p(w|\alpha, \beta)}$$

Unfortunately, the posterior distribution is intractable for finding exact solutions, so many approximate inference algorithms have been used for LDA including variational Bayes, Gibbs Sampling, and their collapsed variants [2, 5, 11, 14, 22].

2.2 Document-Topic Distribution Inference

For our system, we would like to know the topic distribution for unseen documents (documents not yet in the model) without updating the overall topic model. The reason for this is due to computational complexity related to re-computations. Since the topic distribution for streamed documents will be consumed by a recommender system, we need to repeatedly update the topic distribution as more of the document is streamed into the system. For this, we repeated use the variational inference algorithm for LDA presented in [5]. Formally, we are solving the following optimization problem where γ^* and ϕ^* are the variational parameters that are specific to some document w:

$$(\gamma^*, \phi^*) = \arg\min_{(\gamma,\phi)} D(q(\theta, z|\gamma, \phi)||p(\theta, z|w, \alpha, \beta))$$

Specifically, we are interested in γ^* as $\gamma^*(w)$ is the topic distribution representation of document w that we are interested in. Once the document is done streaming, we can then incorporate the entire document into the topic model using an existing online variational inference algorithm presented in [14].

2.3 LDA Example

To explain what LDA accomplishes more concretely, we trained an LDA model against a dataset containg transcriptions from thousands of Ted Talks [18], and we arbitrarily choose to train the model with 50 topics. Each Ted Talk transcription, or "document", is fed into the model in a BOW (Bag-of-words representation). Figure 2 shows the topic assignment probability distribution for a single Ted Talk titled "Alzheimer's Is Not Normal Aging — And We Can Cure It" which helps visualize how each document can be represented by a vector. Looking at Figure 2, there are two prominent peaks at topics 6 and 29, Table 1 shows the most probable words for those topics. Intuitively, it makes sense that a Ted Talk about Alzheimer's disease is associated with a topic centered around general health

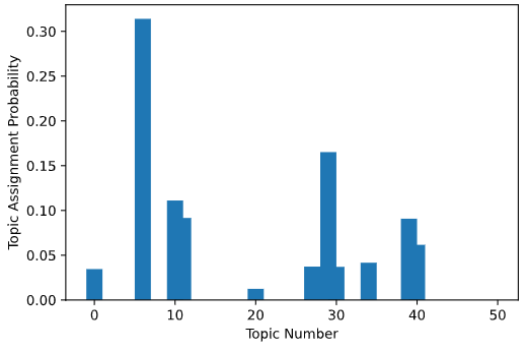

Figure 2: Topic assignment probability distribution for a Ted Talk titled "Alzheimer's Is Not Normal Aging — And We Can Cure It" presented by Samuel Cohen. [8]

and medicine (topic 6) and a topic centered around the brain and biology (topic 29). Additionally, we can better visualize the vector-space of these 50-dimensional topic assignment probability vectors by projecting them onto 2-dimensions using T-SNE (t-distributed stochastic neighbor embedding). Figure 3 shows the T-SNE projection of topic 6 which illustrates some underlying spatial structure which helps us build intuition about how topics can be thought of spatial structures within higher dimensional spaces.

2.4 Characteristics of LDA Systems

There are several key characteristics of LDA topic modeling system that is considered. For one, the document-topic matrix model is typically sparse [19]. This makes most GPU-based topic modeling algorithms very challenging to scale for large text corpora [15]. For CPU-based systems, the sparsity of the model also means directly performing random accesses to memory becomes frequent. As a result, the training speed of an LDA model is determined by the memory performance [6, 19].

Distributed LDA systems [23, 25] are faced with an additional number of tradeoffs. First, similar to distributed data-parallel deep learning systems, there is a high communication overhead due to

Topic 6		Topic 29	
Prob	**Word**	**Prob**	**Word**
.008	disease	.117	brain
.008	health	.020	neuron
.005	care	.014	blood
.005	system	.014	gene
.005	medical	.013	cell
.004	doctor	.013	monkey
.004	death	.009	study
.004	case	.009	body
.004	heart	.008	area
.004	help	.008	rat

Table 1: Most probable terms for topics 6 and 29 in an LDA model with 50 topics trained on Ted Talk transcriptions [13].

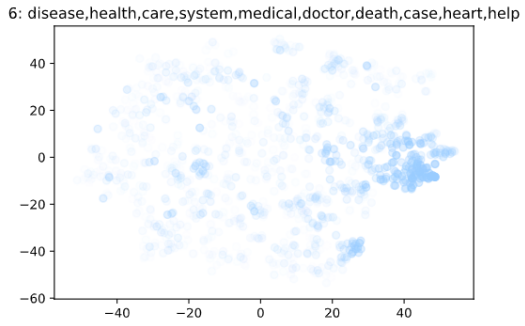

Figure 3: Spatial visualization of topic 6 using T-SNE. 50-dimensional topic assignment probability vector is projected onto 2-dimensions for plotting. Each dot corresponds to a single Ted Talk and its opacity is how probable topic 6 is for that document. The darker the dot, the harder the document is assigned to topic 6.

model synchronization at each iterative model update step. Secondly, the LDA algorithm based on Collapsed Gibbs Sampling (CGS) may require hundreds of iterations to converge. Finally, dealing with straggler workers due to load imbalance can become an expensive underutilization of compute resource. The system aims to minimize the costs of these tradeoffs while benefiting from the scalability and stream processing novelties introduced in this paper.

3 Sampling Experiments

Our research experimented with different techniques to sample a live transcription to approximate the topic distribution of livestreamed content. These experiments were conducted by concatenating multiple TED Talk transcriptions (with different topic distributions) and seeing how the approximated topic distribution changes as the concatenated transcription is incrementally given to the sampler. While the goal of this research is to capture topic changes within a single talk, livestream, or conversation, we tested different sampling techniques against topic changes between talks as a proxy for the performance of the resulting topic distribution approximation.

Figure 4 and Figure 5 are an attempt to spatially visualize how well the approximations capture the intended topic. Each dot in the visualization represented is the topic distribution of a TED Talk projected down to 2-dimensions using T-SNE. The line is the "trajectory" of the topic distribution estimation as each increment is given to the sampler. The color of a single line segment of the topic distribution approximation trajectory corresponds to which TED Talk the current increment given to the sampler belongs to. In other words, if a line segment in the trajectory is red, the TED Talk which contains the transcription for that increment is red.

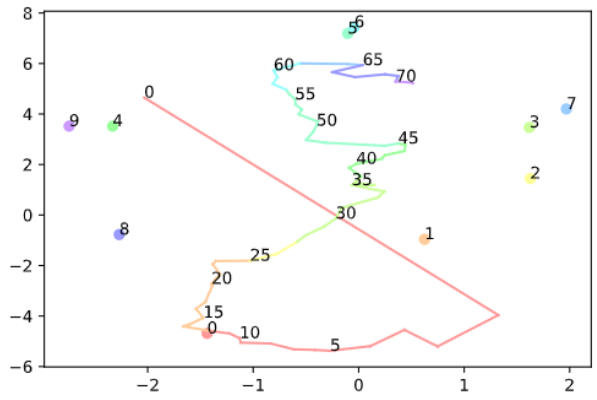

Figure 4: Visualization of topic distribution approximation trajectory using Naive Aggregation. Explanation of visualization in 3.

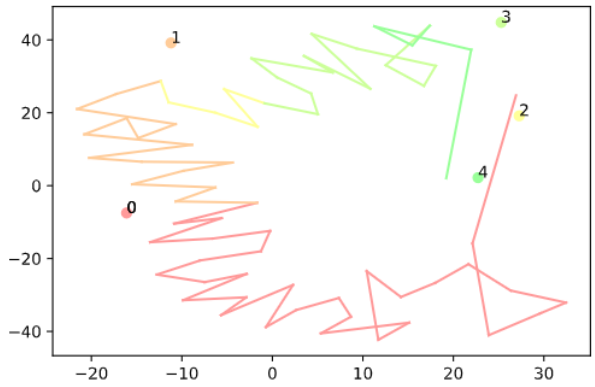

Figure 5: Visualization of topic distribution appoximation trajectory using Static Sliding Window. Explanation of visualization in 3.

3.1 Naive Aggregation

In Naive Aggregation, our sampler simply adds the new words from the current increment to the Bag-of-Words (BOW) representation used for LDA topic distribution inference. While at first, the approximation is able to quickly converge to the topic distribution of the original TED Talk, as more words are added to the BOW representation, the topic distribution gains "inertia". As can be seen in Figure 4, as more increments of the transcription are given to the sampler, the less the topic distribution is able to readily move.

3.2 Static Sliding Window

In Static Sliding Window, our sampler only considers the last N words used for LDA topic distribution inference. From our experiments, this technique appears to work fairly well, and the approximated topic distribution is able to closely converge to the original TED Talk topic distribution. One downside to this technique which can be seen in Figure 5 is that shorter talks were not able to closely converge to the original TED Talk topic distribution due to the window being statically defined (TED Talk 2/yellow in Figure 5).

4 System Design

In this section, we start by introducing the parallelization strategies observed in a distributed topic modeling system. Then we dive further into how workload is partitioned, scheduled and parameter synchronization. Figure 1 maps the system architecture in a diagram.

4.1 Parallelization

There are two types of parallelization. The most common parallelization method is data-parallelism, where the input corpus is partitioned into chunks and distributed across W workers. Each w workers operate on entirely disjoint data chunks of the original corpus. A parameter server holds the contents of the model, and workers must always synchronize with the parameter server fore consistent results. Past work in distributed topic modeling utilize several types of data partitioning strategies in order to parallelize the topic modeling workload.

The other parallelization method is called model-parallelism. Model parallelism is not used often in training systems as they can be challenging and expensive to operate, but hybrid data-model parallelism systems exist. We can do so by partitioning the document-topic matrix and topic-word matrix

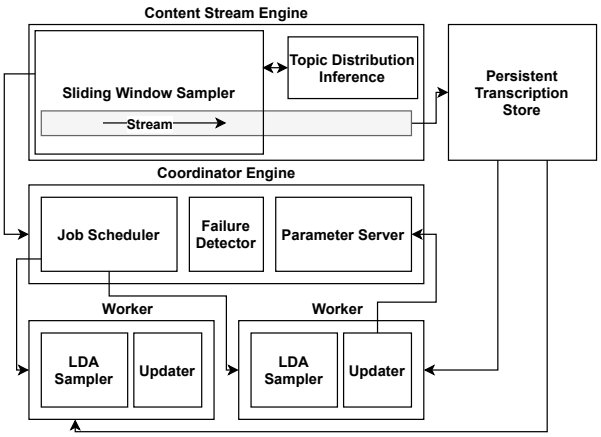

Figure 6: System architecture and logical flow between the topic shift detection module, coordinator engine, and data retrieval from the persistent transcription store.

into a large set of replicas. While model parallelism provides fault tolerance guarantees, there is a significant communication overhead and may hinder the overall performance of the system.

In data-parallel distributed computation systems, load balancing input data is important for optimal performance due to the straggler problem. At every iteration of the training loop, workers must always wait for the slowest worker to complete its job before the coordinator can be ready to schedule the next set of batches. An imbalanced distribution of data could severely impact the potential performance for a system as workers will tend to be underutilized. On a generic distributed computation framework, this can be a challenging problem to solve. Due to the diversity of input data, workloads to train a model tend to be heterogeneous. We focus specifically on co-designing the system to perform well with the characteristics of the data consumed from a non-stationary speech transcription environment.

4.2 Partitioning

There are two common workload scheduling policies used to partition data in a distributed LDA topic modeling system, partition-by-document and partition-by-word.

Algorithm 1: Sliding Window Inference Algorithm

Input : Stream of words W = $\{<w_1,t_1>,<w_2,t_2>,..,<w_{n-1},t_{n-1},<w_n,t_n>,..\}$
Result: Stream segmentation event topic T
D=[], U=[];
while do
  l = time t of end of prior window;
  while *w in W and l < n* **do**
    Check topical diff of word utterances in current set of utterances;
    Concatenate new utterances if related;
    Update window center;
  end
  Insert prior segment into tail of D;
  Update left index with current time t_n from pair;
  Infer topic distribution of current segment;
  Publish new segment to event stream T;
  U=[];
end

Partitioning by document is trivial. The document-topic model is partitioned while the topic-word model is the summation of the all replicas stored in parameter servers. Therefore, only the replicas of the topic-word model needs to be synchronized. Partitioning by words has the opposite side effect, where synchronization of the document-topic model is necessary. However, in the topic modeling paradigm, the document-topic model is typically much larger than the topic-word model. Therefore, it may requires significantly more synchronization in order to be effective. We also explored partitioning data by sentences, which can be determined by pauses in spoken words identified by an Automatic Speech Recognition (ASR) system. However, this would not solve any of the challenges we have listed previously. Furthermore, this level of granularity is not supported by the considerations

raised by the issues with partitioning the corpus by words.

We propose a novel dialogue-aware partitioning policy called partition-by-segment. Human dialogue consists some unique properties that is not expected in other text corpora. The topical shift in a discussion occurs more frequently. This can cause a significant load imbalance in partitioned data, unlike what is expected in a traditional document corpora, where each document focuses on a single idea or theme. Due to the sparsity of topics seen in text corpuses, memory access patterns in LDA algorithms are well researched and reveals sub-optimal performance due to frequent cache misses on the parameters in the model. By identifying drifts in topics, we further improve upon an optimal memory access pattern seen in distributed partition-by-document policies. By partitioning the workload into a set of segments, we can expect better locality in the models, which addresses the problems with sparsity in document-topic matrices.

Maximizing locality can also help the system exploit the eviction properties of an L1 Cache in a CPU or GPU, enabling a much faster cache driven sampling algorithm. Accessing main memory incurs high random access latencies and tend to be quite slow, even in modern computers. This means potentially less random accesses due to the sparse nature of the models. This could provide many significant benefits, especially for an Bayesian inference-based algorithms that are primarily bottle-necked by memory accesses and data synchronization.

In order to leverage the characteristics of the corpus and schedule the accurate data segments, we must identify the drifts that occur in streams of speech transcription. The question remains, how can we identify the drifts that occurs in streams of speech transcription?

4.3 Online Sliding Window

We propose a unsupervised subtopic shift detection module that extracts the streamed data and produces a sliding window of topically similar conversations. A similar strategy has been used in prior text segmentation work [3, 12]. The module will detect the topical changes in a ongoing conversation and help inform the schedulers data partitioning and load balancing policy. The segmentation drift detector will also perform online pre-processing of incoming transcription streams, filtering out stop words and unreadable text.

Algorithm 2: Basic LDA sampler

Input : token θ
Result: sampled data S
compute the sum of sparse doc-topic matrix
compute the sum of topic-word matrix
sample $multinomial(\theta)$

To implement a real-time topic-aware text segmentation policy, we employ an adaptive sliding window algorithm. In our setting, new streams of dialogue and transcription text arrives in real-time as tokens of a word id, document id, and current timestamp t_i. Words and utterances are concatenated into U over time, used to observe for any changes . The topic adaptive sliding window samples and compares word utterances mentioned in a prior set of utterances and observes for any drifts that go beyond a parameterized threshold.

4.4 Scheduling

The scheduling engine in the coordinator will be responsible for scheduling workloads. The stream processing unit will publish drift detection events over an incoming topical stream of data, then prepare the new data segment ready for topic modeling. The coordinator will then be informed that a new segment is pre-processed, queue a new job and inform an idle worker that a new corpora of data is ready to be sampled. The idle worker will request for the specific data chunk from the parameter server, and the stream processing unit will transfer the data into the worker. The worker will then compute the chunk of data, update the document-topic model and the topic-word model replicas, and synchronize the replicas back to the parameter server. The worker will inform the scheduler that the job

is complete, return to an idle state, and await for a new workload to be scheduled onto its machine.

4.5 Parameter Server

The parameter server is designed to handle the sparse nature of the models and the dynamic nature of non-stationary continual learning streams. The parameter server architecture, however, incurs a high communication cost as all workers must directly update and read fresh parameters stored in the parameter server. To minimize some properties of high communication cost, we can implement aggregation algorithms directly on the parameter server.

4.6 Fault Tolerance

We intend to provide mechanisms that deal with machine failures. Machine failures could happen frequently on heavy compute and memory access systems. Furthermore, critical time spent on running the computations could be lost if failures are not handled gently. Our system will monitor worker failures with a separate thread managed by the coordinator that sends a frequent heartbeat to all workers. If workers do not respond after repeated attempts and some timeout value $T_{timeout}$, the coordinator can assume the worker has failed. If a worker encounters failure or error, the coordinator can perform a look up on the details of the failed job allocated to that worker and reinitialize the job on a separate, idle worker.

To account for potential parameter server failures, we will implement a checkpointing step that takes a global snapshot of the model into a persistent data store. However, this process will be parameterized to the scale and frequency of parameter server failure occurences, since the rate at which the models can be snapshotted should be limited as not to incur heavy read or write overhead.

Upon the event of a coordinator failure, workers will recognize that heartbeat messages are no longer being sent by the coordinator using their own $t_{timeout}$ threshold. Workers will immediately stop work and prevent new updates to the parameter servers upon job completion. The coordinator must be restarted with the latest snapshot and prior state of the parameter servers.

5 Evaluation

The implementation is partially complete and the experiments performed below does not utilize the system design in this report. We first evaluated the effectiveness of the topic modeling techniques introduced in this paper. We conducted some theoretical experiments on a single node CPU machine. This is necessary as the system is highly dependent on an algorithm outside the domain of the project. The initial experiments demonstrating topic models are performed on a single core Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz, with a cache size of 30720 KB. For initial steps, we experiment on LDA to profile the spatial locality of topics generated using this method. This data is plotted in Figure 7. The dataset used for this experiment is the TEDTalk dataset [13]. For future experiments, we will use Spotify's 100,000 Podcasts dataset released for TREC2020 as it is the primary objective our project [7].

Next, as presented in Figure 8, we run baseline experiments using Apache Spark's MLlib library, installed on 10 nodes in the cluster. We then evaluate it's scalability by running 5 experiments each on multiple configurations and computing average job completion times. As we see in the figure, this process typically takes around a few minutes, depending on the size of the input corpus, the number of topics, and the size of the dimensions. This, however, is not a comparative baseline with the method and environment we are exploring, as Spark LDA does not natively support a sliding window batching strategy.

We then run a sliding window test across multiple documents within the dataset. It is the same strategy demonstrated in Figure 5, which shows the behavior of the topic distribution changing as topics evolve over time. The sliding window is demonstra-

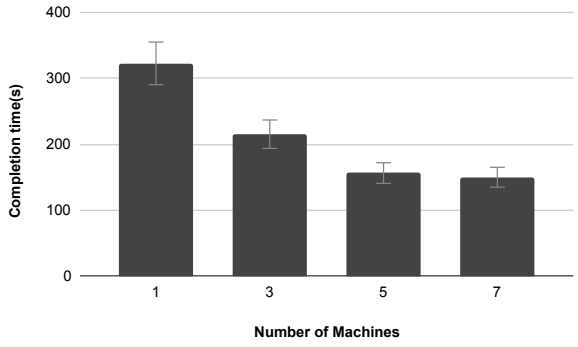

Figure 7: Job completion time to evaluate per-document topics on TED text transcription corpus using baseline LDA with Apache Spark and its rich MLlib LDA library.

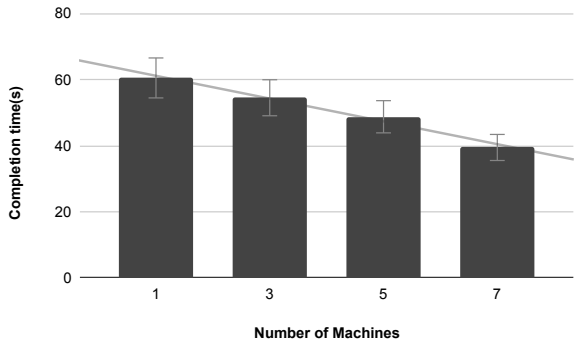

Figure 8: Job completion time for Static Sliding Window, splitting documents in the TED dataset into smaller granular sets of words.

bly faster at less than a minute, but document sizes are much smaller as the sliding window segments are generally smaller than a typical document. However, since the sliding window size may be adjusted dynamically, scaling may not always provide noticeable benefits as the window can be large or small during separate rounds of evaluation. However, splitting the segments into smaller pieces and delivering batches to more machines does provide a slight improvement on a completion of a round.

6 Conclusion

We have introduced a system designed to infer topics within a conversation over large volumes of real-time text transcription data. In this paper, we study various ways topic inference can be done against text data, and current implementations of large scale topic modeling system. We expand large scale topic modeling systems into a streaming environment, where topics are inferred over a time-dependent sliding window rather than per-document. The system is partially completed and will be later improved.

7 Future Work

While we have done some evaluation on several key components, the overall system remains under development. As we continue the experiments on our system design, we may expect to uncover previously undetermined flaws and may decide to adjust some design choices. We will also further evaluate the performance of our system after validating the feasibility of the task.

References

[1] K. S. Arun and V. K. Govindan. A hybrid deep learning architecture for latent topic-based image retrieval. *Data Sci. Eng.*, 3(2):166–195, 2018.

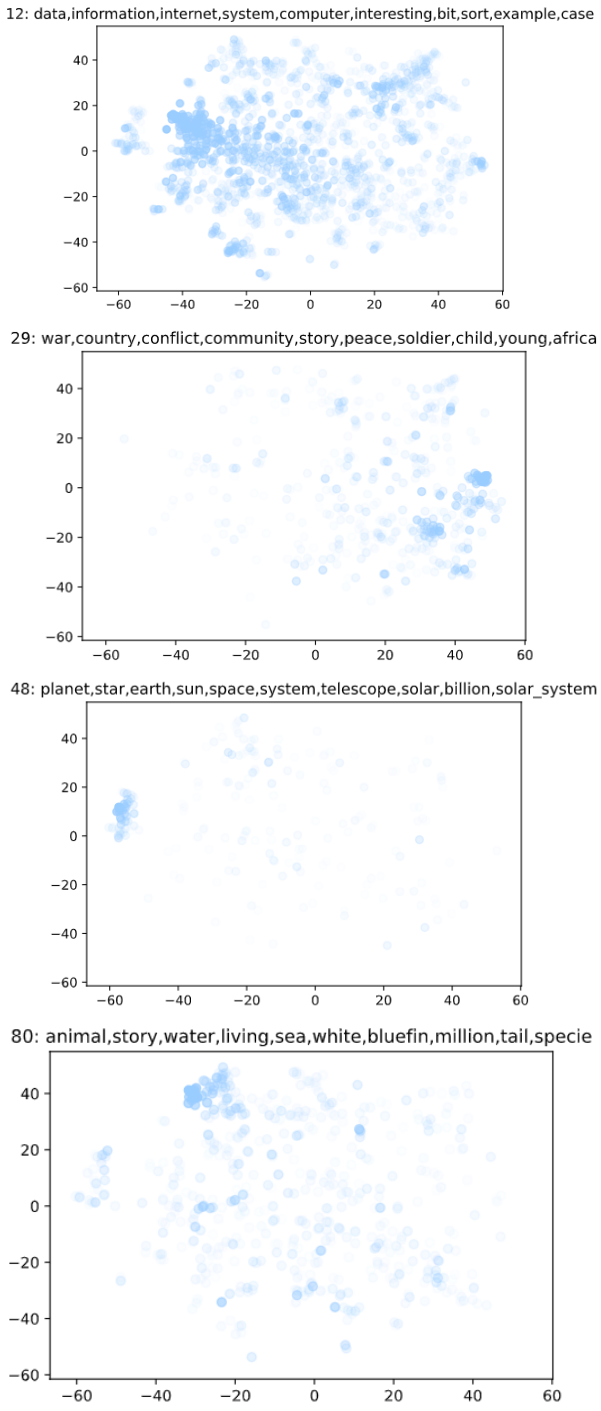

Figure 9: Topics inferred from the document-topic distributions generated from a trained LDA topic model. Here, the 100-dimensional document-topic distribution vector (the LDA model was trained on 100 topics) is projected onto 2-dimensions using T-SNE and the opacity of a dot corresponds to how much a TedTalk corresponds to the topic in the title of each plot.

[2] Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. On smoothing and inference for topic models. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, page 27–34, Arlington, Virginia, USA, 2009. AUAI Press.

[3] Albert Bifet and Ricard Gavaldà. Learning from time-changing data with adaptive windowing. In *In SIAM International Conference on Data Mining*, 2007.

[4] David M. Blei and John D. Lafferty. Dynamic topic models. New York, NY, USA, 2006. Association for Computing Machinery.

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

[6] Jianfei Chen, Kaiwei Li, Jun Zhu, and Wenguang Chen. Warplda: A cache efficient o(1) algorithm for latent dirichlet allocation. 9(10), 2016.

[7] Ann Clifton, Sravana Reddy, Yongze Yu, Aasish Pappu, Rezvaneh Rezapour, Hamed Bonab, Maria Eskevich, Gareth Jones, Jussi Karlgren, Ben Carterette, and Rosie Jones. 100,000 podcasts: A spoken English document corpus. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5903–5917, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.

[8] Samuel Cohen. Alzheimer's is not normal aging – and we can cure it, Jun 2015.

[9] Lan Du, Wray Lindsay Buntine, and Huidong Jin. Sequential latent dirichlet allocation: Discover underlying topic structures within a document. 2010.

[10] Ying Fu, Meng Yan, Xiaohong Zhang, Ling Xu, Dan Yang, and Jeffrey D. Kymer. Auto-

mated classification of software change messages by semi-supervised latent dirichlet allocation. *Information and Software Technology*, 57(Complete):369–377, 2015.

[11] Yang Gao, Jianfei Chen, and Jun Zhu. Streaming gibbs sampling for lda model, 2016.

[12] Marti A. Hearst. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Comput. Linguist.*, 23(1):33–64, March 1997.

[13] François Hernandez, Vincent Nguyen, Sahar Ghannay, Natalia A. Tomashenko, and Yannick Estève. TED-LIUM 3: twice as much data and corpus repartition for experiments on speaker adaptation. *CoRR*, abs/1805.04699, 2018.

[14] Matthew Hoffman, Francis Bach, and David Blei. Online learning for latent dirichlet allocation. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

[15] Kaiwei Li, Jianfei Chen, Wenguang Chen, and Jun Zhu. Saberlda: Sparsity-aware learning of topic models on gpus, 2016.

[16] B. Peng, B. Zhang, L. Chen, M. Avram, R. Henschel, C. Stewart, S. Zhu, E. Mccallum, L. Smith, T. Zahniser, J. Omer, and J. Qiu. Harplda+: Optimizing latent dirichlet allocation for parallel efficiency. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 243–252, 2017.

[17] Shengsheng Qian, Tianzhu Zhang, Changsheng Xu, and M. Shamim Hossain. Social event classification via boosted multimodal supervised latent dirichlet allocation. *ACM Trans. Multim. Comput. Commun. Appl.*, 11(2):27:1–27:22, 2014.

[18] Anthony Rousseau, Paul Deléglise, and Yannick Estève. Ted-lium: an automatic speech recognition dedicated corpus. In *Conference on Language Resources and Evaluation (LREC)*, pages 125–129, 2012.

[19] Khoat Than and Tu Bao Ho. Inference in topic models: sparsity and trade-off, 2015.

[20] Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, page 448–456, New York, NY, USA, 2011. Association for Computing Machinery.

[21] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. *CoRR*, abs/1409.2944, 2014.

[22] Limin Yao, David Mimno, and Andrew McCallum. Efficient methods for topic model inference on streaming document collections. New York, NY, USA, 2009. Association for Computing Machinery.

[23] Hsiang-Fu Yu, Cho-Jui Hsieh, Hyokun Yun, S.V.N. Vishwanathan, and Inderjit S. Dhillon. A scalable asynchronous distributed algorithm for topic modeling. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, page 1340–1350, Republic and Canton of Geneva, CHE, 2015. International World Wide Web Conferences Steering Committee.

[24] Jinhui Yuan, Fei Gao, Qirong Ho, Wei Dai, Jinliang Wei, Xun Zheng, Eric P. Xing, Tie-Yan Liu, and Wei-Ying Ma. Lightlda: Big topic models on modest compute clusters, 2014.

[25] Lele Yut, Ce Zhang, Yingxia Shao, and Bin Cui. Lda*: A robust and large-scale topic modeling system. 10(11), 2017.