MODERNIZING TOPIC MODELS: ACCOUNTING FOR NOISE, TIME, AND DOMAIN KNOWLEDGE

A Dissertation
submitted to the Faculty of the
Graduate School of Arts and Sciences
of Georgetown University
in partial fulfillment of the requirements for the
degree of
Doctor of Philosophy
in Computer Science

By

Rob Churchill, M.S.

Washington, DC
December 3, 2021

MODERNIZING TOPIC MODELS: ACCOUNTING FOR NOISE, TIME, AND DOMAIN KNOWLEDGE

Rob Churchill, M.S.

Dissertation Advisor: Lisa Singh, Ph.D.

Abstract

Data has evolved rapidly since the inception of topic models over twenty years ago. The most popular topic models perform poorly on large contemporary data sets that contain short, noisy texts. This dissertation aims to produce a suite of topic models capable of accurately modeling these new types of data. We begin by tracking the evolution of topic models from inception to modern days. We then propose a flexible preprocessing pipeline that can be adjusted for different levels of noise in the data. The core contribution of this dissertation is the development of a new class of topic models, the topic-noise model. Topic-noise models jointly model topic and noise distributions, greatly increasing the quality of topics derived from social media posts. While static topic models are useful for many settings, they are not well suited for temporal analysis. To identify meaningful topics in this setting, we propose a dynamic topic-noise model that tracks the evolution of topics through time by passing the topic and noise distributions from one time period to the next. Often, researchers have considerable domain knowledge pertaining to the data set being modeled, and wish to infuse their expertise into the topics being generated. For this scenario, we propose a semi-supervised topic model that uses seed topics and selective oversampling to produce a topic set reflective of the user's guidance. We conduct an extensive empirical analysis of all of our models, using quantitative and qualitative methods. All of these contributions culminate in a furthering of the

collective capabilities of topic models, and now topic-noise models, and by sharing the models and code, enhances future research in the field.

INDEX WORDS: Topic noise model, Topic modeling, social media, noise, temporal, generative models, graph models

Acknowledgements

I would like to thank my family and friends for their love and support throughout my life. The love of learning was instilled at me from a young age, and that may have worked a little bit too well, considering I am writing these acknowledgements. I hope that thanks to my defense, you will have learned enough to stop asking me what a topic is for the rest of my life.

I have had the fortune of getting to know the Georgetown undergraduate students on a much closer level than most graduate students. For that, I have to thank the Georgetown club ice hockey team, in particular Chad Heal and Dan Shea, who invited me to join them at the beginning of my masters studies. The hockey players have been a constant source of fun and a the best stress-reliever anyone could ask for. The Hoya hockey team would be unrecognizable without Stan Lechner, whose passion for our team has crossed generations. My friends, linemates, and teammates have made my time at Georgetown a truly unique one. I can't imagine it any other way.

Without one particular person, my experience at Georgetown would have been remarkably different. I may never have pursued a doctoral degree, and I certainly would not have finished one, without my advisor, Lisa Singh. I am extremely fortunate to have someone so encouraging and patient to guide me along the way. She has been an incredible mentor and role model to me both when conducting research and in other aspects of life at Georgetown.

In my time at Georgetown, Lisa has taken on more responsibilities than any other professor that I know of. While I could list off all of these laudable pursuits, the true

mark of her character shines through in one simple fact: despite her many interdisciplinary projects, meetings with provosts and presidents, and collaborations with the United Nations, she still leads the Georgetown Women Coders club, teaching workshops for anyone who wants to learn to code, and helping to prepare women of all levels for careers in her field. Her personal efforts have resulted in Georgetown harboring one of the few computer science departments in the world that can boast a relatively equal distribution of men and women in its undergraduate classes. Miraculously, Lisa has always had enough time and energy to listen to my crazy ideas, quickly understand them, redirect me if I was going off the rails, and help me understand topics that were new to both of us. It was her idea to work on modernizing topic models in the first place, and she has influenced every page of this dissertation.

This work was made possible by the National Science Foundation (NSF), the Massive Data Institute at Georgetown (MDI), and by an Alpha Edison Gift.

Bibliographic Note

Chapter 1 introduces the problem that we aim to solve in the rest of this dissertation. Chapter 2 reviews relevant works in topic modeling and other related fields that we draw on for our work. It is derived from our survey paper *"The evolution of topic models"* [26], co-authored by Lisa Singh. Chapter 3 defines the common notation across the rest of the dissertation, and explains our views on noise words.

Chapter 4 is primarily derived from the paper *"textprep: A text preprocessing toolkit for topic modeling on social media data"* [23], co-authored by Lisa Singh, and from *"The impact of pre-processing classes on meaningful topics from online text data"* [28], co-authored by Lisa Singh and Josh Pasek. In Chapter 4, we describe our attempts to remove noise from social media data using text preprocessing methods. Chapter 4 culminates in well-defined preprocessing classes, rules, and configurations, and a pipeline for quickly iterating over many configurations for a given data set.

Chapter 5 is based on the paper *"Percolation-based topic modeling for tweets"* [21], co-authored by Lisa Singh. Chapter 5 details the construction of our λ-CLIQ topic models. These models, which come in augmented and basic variants, are graph-based models that leverage ngrams to find noiseless topic kernels with a word co-occurrence graph.

Chapter 6 is based on the paper *"Topic-Noise Models: Modeling Topic and Noise Distributions in Social Media Post Collections"* [22], co-authored by Lisa Singh. Chapter 6 defines topic-noise models, a new type of topic model that jointly models topic and noise distributions. Chapter 7 is based on the paper *"A Guided Topic Model*

for Social Media Data," co-authored by Lisa Singh, Rebecca Ryan, and Pamela-Davis-Kean, which is under review at the time of this writing. Chapter 7 allows for users to provide seed topics and give feedback to a semi-supervised generative model that leverages a topic-noise model to filter noise from topics. Chapter 8 is based on the paper *"Temporal Topic-Noise Models for Social Media Data Sets,"* co-authored by Lisa Singh, which is under review at the time of this writing. Chapter 8 creates dynamic topic-noise models for tracking topic evolution through time. Chapters 9 and 10 present future work in the field of topic models and our conclusions.

Table of Contents

List of Figures

List of Tables

"It's hard to crab without a boat."

To my mom, Sheri, for teaching me to love to learn

To my dad, Rob, from whom I learned to never give up

To Katherine, who taught me to share

To Alex, my partner in crime

To Mallory, my partner in everything else

To Nanny, for the fuel

And to Bop, for the boat

Chapter 1

Introduction

One of the most important distinctions between humans and other forms of life is the ability to communicate efficiently. Language has evolved for tens of thousands of years, from small communities to countries and regions. The ways that we communicate have evolved as well. Fireside stories gave way to town squares, town squares to telegrams, telegrams to radios and telephones, and most recently, the internet. The most efficient way to store information was originally through our own brains, through passing down stories around the fire. Eventually, humans learned to record information on parchment, compiling books of knowledge through highly-trained scribes. With the invention of the printing press, the speed at which we could record information grew exponentially, and the volume of information that we could store grew exponentially with it. Five-hundred years after the invention of the printing press, the mechanical computer sparked another period of exponential growth, allowing the storage of more and more information on smaller and smaller pieces of silicon. With the addition of the internet, the world's most powerful connector of people to date, the computer has allowed humans to communicate and create information at a rate unparalleled in the history of humankind. This leaves us with a burning question. What on Earth (or off) are they talking about?

In the age of the printing press, the publishing of books and newspapers were limited by the speed and number of the presses themselves. These texts could be sorted into categories and subcategories to enable readers to quickly and easily understand

their content, and at a higher level, what content was important and relevant in their time. This is no longer true. According to the International Publisher's Association and World Intellectual Property Organization, just over 4.7 million books were published in the entire year of 2018 [4].[1] In contrast, over 500 million tweets [40] and 420 million Facebook status updates [72] are published per day. Social media posts are a new type of document and topic models can be used to understand the themes of online conversation. Unfortunately, these documents come with their own challenges. 1. *Document length*: Instead of well-edited books and articles, many text documents are short, unedited social media posts containing very few words. In the case of Twitter, tweets are restricted to 280 characters. 2. *Sparsity*: Instead of texts consisting of thousands of carefully edited words and phrases, social media posts contain a few dozen hastily typed words, often accompanied by a hashtag and a URL. Reinforced word co-occurrence patterns within social media posts simply do not exist at the rate that they do in longer texts. This lack of strong word co-occurrence can result in noisy, incoherent topics when topic models not designed for sparse, short text are used with social media data. 3. *Volume*: Instead of the thousands of books and research papers being published every year, there are millions of social media posts generated every single day. Many of the original topic models were built on statistical models supported by intractable inference problems that required relaxation just to be feasible for thousands of documents. 4. *Rate of Change*: Instead of information being published on a yearly, quarterly, monthly, or even on a weekly basis, social media facilitates the publishing of information in real time. All of these factors culminate in the realization that it is impossible for humans to manually read, categorize,

[1]This figure was calculated by summing the number of books registered for an ISBN in 2018.

or understand all or even most information being produced today. As such, we need a way to automate this process.

Topic models are the answer to this problem. Created initially to categorize text documents for large online databases like JSTOR, these models work by learning the co-occurrence patterns of words within a document in order to produce a distribution of topics that the document covers. Topic models have been utilized in many different disciplines to understand different types of texts – from tweets to books. Researchers have applied topic modeling to understand themes from historic newspapers [69, 105]. Sleeman et al. [89] incorporated topics models in their analysis of the evolution of climate change literature. Jocker and Mimno used topic modeling to identify themes in 19th century literature [45]. Ryan et al. [83] used topic models in a semi-iterative approach to discerning the most popular parenting topics on Twitter, while Bode et al. [14] use a similar approach for understanding the 2016 US Presidential election using newspapers, tweets, and surveys with open-ended responses.

Topic models can also be incorporated with other learning models, to increase predictive power and interpretability. Singh et al. [87] use topics as input into hierarchical Bayesian models to help predict forced migration of displaced persons. Topic models have also been adapted to help predict the effects of genetic variants [5]. They have been used to generate tags for content tagging systems [49], to organize online recommendation systems [1], and to explain latent factors that lead to recommendations [82]. They have also been used to improve sentiment analysis in text [46, 58]. Topic models are a useful tool in information retrieval, aiding in query expansion and document smoothing [107, 108]. What should be apparent from these examples is that topic models are being used broadly, both as a final result and as a feature into other predictive models.

Traditional topic models, such as Latent Dirichlet Allocation (LDA) [12], work well for long, well-edited documents such as books, research papers, and newspaper articles. However, traditional topic models are not easily scaled to the order of millions of documents [38, 90, 106], and they often perform poorly on social media data due to the short length of the text and the levels of noise present in them [23].

Topics themselves are subjective by definition. What one person sees as a perfect topic may not encompass another facet that someone else feels should be included. Another person might think that the same topic is too broad, and should be split into two. There is no single "correct" topic set for a data set. However, there are some topic sets that are more interpretable than others. Therefore, our aim is to produce topics that can be easily identified and understood by humans. Figure 1.1 shows some topics from the 2016 United States Presidential Election that have varying noise levels, leading to varying levels of interpretability. The top-left topic contains stopwords like 'of' and 'its', along with other uninformative words like 'their' and 'told.' The top-right and bottom-left topics contain misspellings, punctuation, hashtags words, and other uninformative words like 'hey' and 'need.' In the bottom-right, we see a better quality topic, which contains words referring to taxes and the refusal of Donald Trump to release his tax returns. While we cannot say for sure that any of these topics are wrong, we can show, through qualitative and quantitative evaluation, that certain topics are more easily interpreted than others.

There are few topic models that are capable of modeling topics as they evolve over time [10, 34, 101]. Volatility of topics over long periods of time can lead to confusion in the interpretation of topics. Being able to track the evolution of topics would improve the understandability of topic sets generated over a long period. Finally, users of topic models often possess knowledge specific to the domain of the data set that is

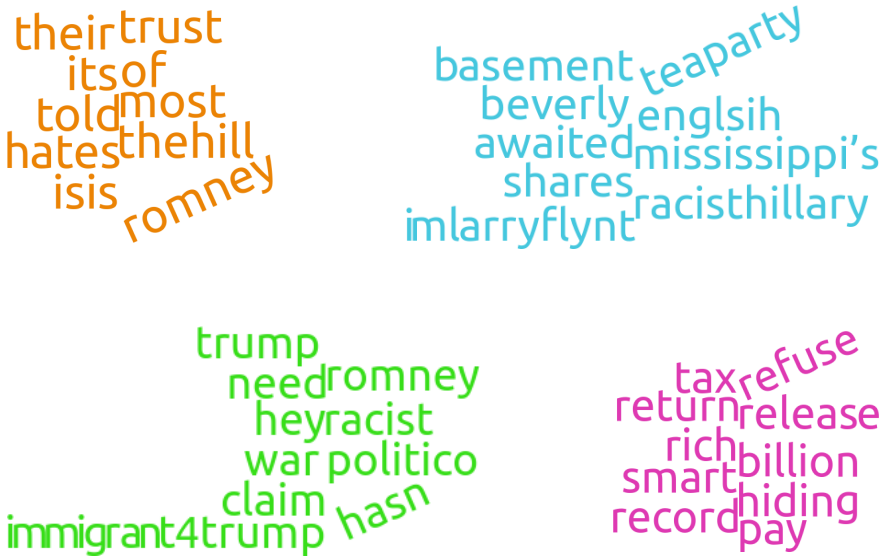


Figure 1.1: Topic Word Clouds

being modeled, and there is very little that can currently be done to leverage that knowledge to produce better topics.

With these problems in mind, we set out to create a topic model, or set of topic models, that addresses them. We aim to create a suite of topic models that is robust to noise, can track topics on a temporal plane, and can incorporate domain knowledge; in short, the holy grail of topic models. On our quest for the holy grail (of topic models), we define different types of noise in the context of topics, and we find that not all evaluation metrics are created equal. We propose new evaluation metrics and compare them to past methods, and we show the value of qualitative evaluation alongside quantitative evaluations. Our quest leads us not to the holy grail that we were looking for, but to a new type of model – the topic-noise model. A topic-noise model jointly models the underlying topics and noise distributions of a data set, provides less noisy topics, and can be used in conjunction with traditional topic models to create near-noiseless topic sets. We propose a semi-supervised topic model that takes into account the guidance and feedback of the user in the form of 'seed

topics,' and uses a topic-noise model to filter noise from the final topics. Finally, we create a dynamic topic-noise model to allow for the generation of noiseless topics on a temporal plane.

The contributions of this dissertation are as follows: (1) We address the problems of the current state of topic models when it comes to modeling on social media data and other noisy sources. (2) We define noise in the context of topics, and propose a graph-based topic model that gracefully removes noise. (3) We show the value, and limitations, of data preprocessing in terms of topic model performance. (4) We propose a new type of model, the topic-noise model, that is designed to jointly model topic and noise distributions in social media data sets. (5) We propose a dynamic topic-noise model capable of performing temporal modeling. (6) We propose a semi-supervised topic model that allows for user guidance and feedback that uses a topic-noise model to filter noise from guided topics. (7) We propose and identify novel evaluation methods for topic modeling, both qualitative and quantitative. (8) We show through experimentation and the use of our qualitative and quantitative evaluation methods that our models outperform state-of-the-art topic models for their respective tasks. (9) We release our models, code, preprocessing pipeline, and evaluation metrics that we use to do all of this so that others may use it in their own research.[2]

This dissertation is organized as follows: Chapter 2 provides a thorough background of previous topic models as they have evolved into the models we use today, contemporary topic models related to our research, and commonly used evaluation methods for topic modeling. Chapter 3 defines the notation used throughout the rest of the dissertation, and describes in detail the quantitative methods that we use to evaluate topics. Chapters 4-7 detail the major contributions of the dissertation. Future work is presented in Chapter 9, and conclusions are presented in Section 10.

[2]Our code can be found here: https://github.com/GU-DataLab

Chapter 2

Related Literature

In this section, we delve into the history of topic models and describe in depth some of the more popular topic models [26]. The topic modeling papers here have had a vast influence on the creation of our own models, whether directly or indirectly, positive or negative. The first part of this section is devoted to evaluation metrics that are used throughout different papers. In the second part, we describe the models themselves and how they have evolved from their roots up to the present.

2.1 Evaluation of Topic Models

An important part of topic modeling is evaluation. In fact, an entire survey could be written just examining evaluation methods for topic models. Many different groups have conducted research on the effectiveness of certain evaluation methods [19, 68, 96].

In this section, we take a look at some of the more popular evaluation methods for topic models. We focus on those that have been used by more than one approach we present, and organize them into three general categories: evaluation of coverage, evaluation of coherence, and qualitative evaluation.

2.1.1 Coverage

Coverage refers to how well the concepts in the document collection are represented. Coverage can be divided into two types of coverage, topic coverage and document

coverage. Topic coverage measures assess how well topics are discovered, i.e. are the topics in a document collection identified by the model. The most prevalent topic coverage measure is *topic recall*. Topic *recall* is the fraction of ground truth topics recovered by the topic model.

Document coverage measures evaluate how well documents are represented by topics. Topic model *accuracy* is a typical measure for evaluating document coverage. It is defined as the fraction of documents that are accurately labeled by the topic model. In order to evaluate topic recall and accuracy, ground truth topics must be available. As will be seen, topic recall is used more sparingly than topic accuracy.

When a full set of ground truth topics are not readily available, other metrics are used. *Purity* is the accuracy of the model if documents are always assigned the dominant topic. This metric attempts to penalize models that assign a large number of low probability topics to documents, as opposed to a model that assigns a high probability to a single topic across the document collection.

Another way to measure convergence is to compare topics across topic models. KL-Divergence is used to show how a new model's probability distribution over topics differs from that of a baseline topic model. *KL-Divergence* is the expectation of the log difference between the underlying probability distributions of the topic models. Instead of relying on ground truth topic sets, KL-Divergence allows one to compare the topic set of a new model to a previously established state-of-the-art model.

All of the coverage methods described so far require, to some degree, prior knowledge topics, either in the form of ground truth topics or in the form of a state-of-the-art topic model's topic distribution. In 1999, Hofmann took a different approach when he introduced perplexity [39]. Instead of relying on ground-truth data to compute topic or document coverage, perplexity is a probabilistic measure of how well a model can predict a held out sample of documents. It is a way to compare probabilistic

topic models. More precisely, *perplexity* is the average negative log-likelihood of held-out documents. A lower perplexity means that the topic model is better at labeling held-out documents, and has a better document coverage. Others have used the log-likelihood of held-out documents (not negated) as an evaluation method. It should be noted that a debate exists as to whether or not perplexity is a reasonable evaluation method for determining topic quality [19].

2.1.2 Coherence

While coverage is an important measure for high quality topics, by itself, coverage is not sufficient for guaranteeing individual topic quality. To combat the potential for noisy and unintelligible topics, authors turn to what we call *coherence* evaluation methods. These methods attempt to discern the usefulness of individual topics, and of words in individual topics.

Many of the earlier works took a classic approach to coherence evaluation – *precision*. Given ground truth topics, and the top words from each approximated topic, a precision score can be calculated by taking the fraction of words from each approximated topic that falls into the best-fitting ground truth topic.

As we will see, coherence evaluation methods were largely abandoned after some of the initial topic models were introduced. Only recently has coherence resurfaced as an important component of topic model evaluation. The most common coherence metric is *pointwise mutual information (PMI)*. There are many different variants of PMI, but at its core, PMI attempts to measure the closeness of words in each topic based on their relative cofrequencies with each other. Its popularity has increased because unlike precision, no ground truth topic set is required to calculate PMI.

A number of other coherence measures have been used to evaluate topic models. C_V, as defined by Röder et. al [81], is a combination of cosine similarity and normalized

PMI under a sliding window of words in a document. It attempts to capture proximity between words as well as the mutual information and vector similarity. *Diversity* is an evaluation method that attempts to account for word overlap in topics. If a model cannot decide which topic a word belongs to, it may assign it with reasonably high probability to multiple topics. This makes a topic set less coherent. Topic Diversity is the percentage of unique words in the topic set, considering the top-k words of each topic. Like PMI, diversity does not require knowledge of ground truth topics.

Adjusted Rand Index (ARI) measures the agreement of topic-document classifications. For a pair of documents, either they should or should not be clustered together based on some similarity measure. The Rand Index calculates the percentage of document pairs that are correctly placed together or correctly not placed together. ARI corrects the Rand Index for chance. ARI is most commonly used when the topic-documents labels are known, so that the similarity measure is whether the documents actually belong to the same topic. However, ARI could also employ a similarity measure that does not rely on ground truth labels.

Signal-to-Noise Ratio (SNR), similar to precision, uses ground truth topics to compare the number of correct and incorrect topic words in each approximated topic. A high SNR indicates more coherent topics, while a low SNR indicates high volumes of noise and therefore less coherent topics. Finally, *word Intrusion*, defined by Chang et. al [19], is a human-judged coherence method that consists of giving a human six words, five from the same topic, and one that does not belong in the topic. If humans can consistently pick which word does not belong, then the topic model is judged to be more coherent.

2.1.3 Qualitative

While coverage and coherence can be precisely defined, due to its nature, a qualitative evaluation of a topic model cannot. As such, coverage and coherence evaluations are insufficient judges of a topic model. A *qualitative evaluation* of a topic model is any evaluation that displays the topics produced by a model for readers to assess themselves [11]. Qualitative evaluation includes little to no math or statistics, consisting of a display of topics generated during the experiments. In some cases, a qualitative evaluation is used to compare a single topic across all of the models tested. In others, it is used to show the diversity of an entire topic set generated by the new model. In the case of some temporal models, a qualitative analysis shows topics along with when they are most prevalent along a time scale. A qualitative evaluation of a topic model is used to portray what math and statistics often cannot: human understandability. A qualitative analysis can be misleading if authors choose to display only a small subset of the best topics. However, a well-done qualitative analysis can be very valuable if it helps readers to understand the quality of topics generated and/or how a new model's topics might differ from baseline models.

2.1.4 Evaluation Mapping to Proposed Models

Table 2.1 shows which models described in this section used which evaluation methods for their experiments. Table 2.1 contains only the evaluation methods that were defined and used in more than one paper examined in this survey. Many more evaluation methods have been used in the past to demonstrate the quality of topic models. While our list is not exhaustive, it does show a trend in methods over time. Coverage and coherence methods have waxed and waned in popularity throughout the years,

with perplexity falling in favor of pointwise mutual information. The qualitative evaluation has become just as necessary as a good quantitative evaluation for portraying topic model quality. Other evaluation methods find a home here and there, but PMI and qualitative evaluation have become common practice.

2.2 The Birth of Topic Models

2.2.1 Precursors to Topic Models: Latent Semantic Indexing

Topic models can trace their origins back to 1990. In their paper *Indexing by Latent Semantic Analysis*, Deerwester et. al described how one could use latent semantic analysis to automatically index and retrieve documents from large databases [31]. The authors devise a model called Latent Semantic Indexing (LSI).

Given a vocabulary, a set of documents can be represented by a word-document matrix, where each document is a vector the size of the vocabulary, and each entry in the vector is the frequency of the relative vocabulary word in the document. LSI takes the word-document matrix and performs a singular value decomposition (SVD) to reduce the dimensionality on the document-side of the matrix but retain the meaningfulness of the words. In doing so, LSI created the first topic sets from documents. These topics were vectors of word frequencies that were derived by the SVD, and could be compared to the specific word frequency vectors of individual documents in order to classify the documents into topics. Notably, LSI defines what will come to be known as the bag of words model. The bag of words model is oblivious to the ordering of words in a document - it cares only about the frequency of words.

In 1999, Thomas Hofmann kicked off what turned out to be the field of topic modeling. Hofmann introduced Probabilistic Latent Semantic Indexing (pLSI), in an attempt to produce more consistently accurate results in domain-specific documents

Table 2.1: Topic Model Evaluation Methods by Model.

Evaluation Method Prevalence											
	Coverage						Coherence				Qualitative
Model	Recall	Accuracy	LL	Perplexity	KLD	Purity	Precision	PMI	Diversity	ARI	Qualitative
LSI [31]	x						x				
pLSI [39]				x			x				
DMM [71]	x						x				
LDA [11]				x							x
HDP [91]				x							x
CTM [51]			x	x							x
DTM [10]			x								x
TOT [101]		x			x						x
SWB [20]				x			x				x
Online LDA [6]								x			x
Online LDA [38]				x							
Online HDP [98]			x								x
MTTM [67]				x							x
cDTM [97]				x							x
NMF [86]		x									
NMF [47]	x	x					x				
NMF [104]		x						x		x	x
ETM [18]											x
TS [30]		x									
TFM [27]	x	x					x				x
PTM [21]								x	x		x
BTM [103]						x		x		x	x
SATM [79]		x				x		x			
LF-DMM [70]						x		x			x
LF-LDA [70]						x		x			x
NVDM [62]	x	x		x	x		x	x			x
lda2vec [66]											x
PTM [112]	x	x						x			x
ETM [77]								x			x
GSDMM [109]								x		x	
GPUDMM [54]		x						x			
GPUPDMM [54]		x						x			
DREx [8]	x						x	x			x
CSTM [56]		x				x		x			x
WELDA [17]								x			x
LapDMM [57]		x						x			
CluWords [93]								x			x
CluHTM [94]								x			
ETM [35]								x	x		x
D-ETM [34]								x	x		x

Ordered by appearance in this survey.

than LSI [39]. Hofmann never actually refers to topics in his paper, calling them *factors* instead. By replacing the SVD with a generative data model, called an *aspect model* in the paper, he was able to use an expectation maximization algorithm to train the model. Instead of topics being the result of an SVD, the topics were a probabilistic *mixture* of words based on their joint probabilities with documents.

Aside from introducing the probabilistic generative model for topic modeling, Hofmann also introduced the evaluation method *perplexity*.

2.2.2 DMM and LDA: The First Topic Models

DMM. Originally interested in improving the accuracy of text classifiers that until then had used labelled data to classify text, Nigam, McCallum, Thrun, and Mitchell set out to determine how to incorporate unlabelled data into text classification [71]. They would again use expectation maximization along with a generative model, but they added the Dirichlet distribution to the conversation. In their algorithm, a document is sampled by one topic and generated according to that topic based on a prior distribution represented by a Dirichlet distribution. Their experiments show that while in some cases their model improves results over labelled data by 30%, the model can hurt performance in other cases. While originally not framed as a topic model, this model would come to be known in the topic modeling literature as the Dirichlet Multinomial Mixture (DMM).

LDA. The term *topic model* was coined in 2001 by Blei, Ng, and Jordan when they proposed Latent Dirichlet Allocation (LDA) [11] [12]. The authors identified problems with the pLSI model, including the large number of parameters needed to fine-tune pLSI, and the inability of one to query the pLSI model with new unseen documents. Furthermore, the authors of LDA made a key observation that enabled

them to be more successful in finding accurate topics: documents are not generated according to only one topic.

LDA borrows from the ideas of pLSI and creates its own generative model, this time based on the Dirichlet distribution, which they borrow from Nigam et al. [39, 71]. It uses the same bag of words model as pLSI, and the same document-term matrix defined by pLSI. The key innovation of LDA over DMM and pLSI is that LDA does not sample one topic per document. Instead, it repeatedly samples multiple topics per document. This intuitively implies that a document is not generated from only one topic, but from many topics, each with a different probability relative to the words in that document. LDA's goal is to find the parameters of a topic/term distribution that maximizes the likelihood of documents in the data set over k topics. Each document has a probability distribution over the topic set that is proportional to the probability distribution of each of its words over the topic set. LDA uses expectation maximization to train its generative model, sampling multiple topics for each document, leading to a mixture of topics for each document. LDA has two main parameters, α and β, that are used to fine-tune the model. α corresponds to topics per document ratio. Setting α higher results in more topics per document. β corresponds to words per topic ratio. Setting β lower results in fewer words per topic.

The authors show that LDA performs better in terms of perplexity than pLSI. The main data set is a corpus of 16,000 newspaper articles from the Associated Press, containing a vocabulary of 23,075 terms. The authors identify a problem of term sparsity, writing, "the large vocabulary size that is characteristic of many document corpora creates serious problems of sparsity. A new document is very likely to contain words that did not appear in any of the documents in a training corpus" [12]. The authors attempt to account for this by using Laplace smoothing, essentially assigning some very low probability to every word so that it can be classified by the model. The

problem of sparsity persists all these years later in even more extreme fashion due to the vast amount of slang and lack of structure in social media.

LDA is still synonymous with topic modeling, and many other models to this day are variations of this model that was originally conceived almost twenty years ago. LDA has many variations and has been applied to many different tasks in and around the natural language processing realm. Jelodar et. al take an in-depth look at LDA specifically, and describe many of its variants in detail in their survey of LDA [44].

2.2.3 Early LDA Variants

In the years after designing LDA, researchers began to improve upon this model in order to solve some problems not addressed by LDA.

Hierarchical Dirichlet Process. The first problem was that LDA had one main parameter, k, representing the number of topics that should be sampled. This was a problem because most people do not know how many topics an unexplored data set contained. The process of finding the right k for a new data set was tedious, requiring users of LDA to test using numerous settings of k, manually evaluating the results after each iteration. Teh, Jordan, Beal, and Blei introduced the Hierarchical Dirichlet Process (HDP), a non-parametric adaptation of LDA that approximates the parameter k using a probabilistic model [91].

In HDP, the Dirichlet processes representing topic distributions are distributed according to another Dirichlet process. Essentially, not only are topics sampled probabilistically, the number of topics is also sampled probabilistically. HDP still allows for the fine-tuning of α and β in the same way that LDA does, but the k value is chosen probabilistically so as to take the tedious process out of the user's hands. Due to the extra layer of Dirichlet processes, HDP is even more intractable to compute

precisely. The authors replace the deterministic expectation maximization algorithm from LDA with a Gibbs sampler to improve its efficiency.

In their experiments, the authors show that HDP provides topic sets with consistently low perplexity for a nematode biology abstract data set (5,838 abstracts), and 1,447 *Neural Information Processing Systems* (NIPS) articles.

Correlated Topic Models. Another limitation of LDA is that it assumes independence on topics, where in the real world, topics are often correlated with each other. For instance, topics about weather and sailing are probably more correlated than topics about weather and video games. Blei and Lafferty introduce a correlated topic model (CTM) to address this issue [51].

The key difference between CTM and LDA is that in CTM, the authors replace the Dirichlet distribution with a logistic normal distribution, and add a covariance matrix between topics to model correlation. The generative process is identical to that of LDA; the only thing that changes is the distribution from which topic probabilities are drawn. The assumption that exists with a Dirichlet distribution - that topics are independent of each other - is not necessary in CTM, since the covariance matrix of the logistic normal distribution provides a measure of how close topics are to each other. This allows for the model to pick not only words based on topics that are drawn from the observed document, but also from topics that are correlated with the probable topics.

To demonstrate the effectiveness of their approach, the authors perform topic modeling on articles from the journal *Science*. The domain of *Science* articles is spread across many fields of study, but intuitively there should be some correlation over many of these fields. Again, the authors use perplexity as their evaluation method. The authors show that the CTM is able to maintain strong performance even as the number of topics grows, whereas LDA falls off at around thirty topics.

Dynamic Topic Models. LDA captures the topics of a static data set at one moment in time, but it does not capture the evolution of topics across time. Blei and Lafferty create the first temporal topic model, called Dynamic topic model (DTM), with the goal of capturing *topic evolution* [10]. A *temporal topic model* is a topic model that adds a time-based component to topics. Instead of a topic consisting of a set of words, a temporal topic model produces topics that consist of a set of words and a timestamp or time range. A simple temporal topic model could separate documents into a range of dates and perform modeling on each subset of documents in turn.

The authors again use the journal *Science*, which has been published continuously since 1880. As technology and the knowledge of humans has advanced, so too have the topics covered in the journal, so the authors hypothesize that if DTM is a good model, it should be able to capture the evolution of topics throughout the history of *Science*, visualizing the birth and death of topics.

To create the dynamic topic model, the authors remove the assumption from LDA that documents are completely exchangeable. Instead, they slice the data set into time periods, and only inside a given time period are documents exchangeable. The authors replace the Dirichlet distribution of topics with a sequence of Gaussian distributions, each of which represents the topic distribution for a different time slice. The authors state that their Gaussian distributions are an extension of the logistic normal distribution to deal with time series data. The approach that the authors use is not so much a temporal topic model as it is a series of topic models tied together in sequence. The topics for time slice t are conditioned on the topics from time slice $t-1$, and by the documents in time slice t. This leads to topics that seemingly evolve through time, appearing as they become relevant and disappearing as they lose popularity.

The authors show how topics related to Darwin and Einstein spike as each become famous for their work, and slowly die out as time passes on. Another example that they point out in the paper is the topic about the moon, which is moderately popular throughout the late 1800s and early 1900s, but which spikes during the space race of the Cold War. The authors further show the ability of DTM to predict future topics given its current topic distribution. They compare this predictive power to the approach of chaining LDA through the same time slices, and show that DTM is far more accurate when tasked with predicting future topics.

In 2006, Wang and McCallum proposed a temporal topic model based on LDA that worked on a continuous-time model [101]. The advantage of this approach, called Topics over Time (TOT), is that events over time are not uniformly distributed, and therefore uniformly distributed time epochs do not fully capture the evolution of topics over time. The authors model time alongside word co-occurrence in order to capture time-sensitive events, such as different wars and the rise of technology. The continuous distribution over time allows for the user to not specify the length of an epoch, instead relying on the algorithm to find significant word co-occurrences over relevant time periods.

Wang and McCallum mostly provide a qualitative analysis of their topic model, but perform a small quantitative analysis of their model compared to LDA, where they use KL-divergence to show that TOT produces slightly more distinct topics than LDA.

DTM and TOT represent the first attempts at applying LDA in a temporal fashion in order to capture the evolution of topics.

Topic Models Considering Background Noise. In 2007, Chemudugunta et al. [20] proposed a topic model, SWB, that incorporated two secondary distributions to capture special words (specific to a document) and background words (what we

would call noise words). The goal of the proposed method is to capture the difference between generic documents, highly specific documents, and those in between. In theory, this should produce documents that are true to the approximated topics, but that also vary in specificity as is the case in the real world. They use LDA [12] as the basis of their model, and insert the special words and background words distributions directly into the scheme. The authors propose using a random variable to act as a switch, which determines if the next word in a document is generated from the special words distribution, the background distribution, or the chosen topic. The switch variable is sampled from a document-specific multinomial conditioned on a Dirichlet prior, so that every document is individually tailored. The special words distribution is sampled from a document-specific multinomial conditioned on another Dirichlet prior, such that each document has its own set of special words. The background words distribution is sampled from a data set-specific multinomial conditioned on another Dirichlet prior. This final distribution is data set-wide so that commonly used words can be identified across all documents.

In their experiments, the authors show the effectiveness of the background distribution on four different data sets, and compare their model variants (containing both distributions, or just the special words distribution) to LDA using perplexity and precision. The four data sets used in the experiments are of unspecified size, but consist of NIPS articles, Associated Press articles, U.S. Patents, and Federal Register articles, respectively. The authors show that their model variants beat LDA in terms of perplexity, and beat LDA [12] and LSI [31] in terms of precision when tasked with retrieving documents containing at least one query word (a downstream information retrieval task). The authors use a newspaper data set consisting of 3,104 articles about Enron from the New York Times to perform a qualitative analysis, highlighting topic words, special words, and background words to.

While the authors clearly intended their model be used for information retrieval purposes, their proposal is the first real attempt to filter noise words from topics. We believe that they were ahead of their time in this regard. As we will see, identifying and filtering noise in topic models becomes crucial as the type of documents that we care about transitions from long, well-edited documents to short, noisy social media posts.

2.2.4 Computational Complexity of Generative Models

The early implementations of LDA and its descendants were very computationally expensive. Sontag and Roy proved the complexity bounds of generative topic models [90]. They showed that if a document is generated by a mixture of the entire topic set, inference is NP-Hard. However, if a document is generated by a constant number of topics strictly smaller than k, the number of topics, inference can be solved in $O(N^4)$, where N is the number of documents in the data set. These complexity bounds severely limit how fast topics can be approximated in some of the most popular topic models. While approximation algorithms do exist, improving the efficiency of generative topic models is still an important research direction.

2.3 Temporal Topic Models

As discussed in Section 2.2, by 2006, Blei and Lafferty [10], as well as Wang and McCallum [101], had already been experimenting with ways of detecting topics as they rise and fall over the course of time. While their approaches were relatively successful, they were computationally costly.

In 2007, Nallapati et. al proposed a new generative model, MTTM, to improve on the Dynamic Topic Model's temporal aspect [67]. The authors replace LDA's

multinomial distribution over words with a Poisson distribution over the words. For each word in each epoch, an expected frequency in each topic is kept in the form of a Poisson mean. The authors represent each topic as a vector of Poisson means, which allows for the evolution of topics over time epochs to be monitored directly. The authors show that MTTM works in a hierarchical manner, starting with the smallest time epochs, and building up into bigger and bigger epochs. This hierarchical structure allows for users to "zoom in" on certain time periods for a particular topic, getting more and more specific as one drills down.

In 2008, Wang et. al proposed cDTM, a continuous version of the DTM. In it, as in TOT, cDTM does not require time to be discretized, opting for a continuous distribution that is less computationally intensive that the orginal DTM. The authors first probabilistically select a time frame for each topic before inferring the word distribution. This approach allows for faster inference of a more fine-grained timeline of topics.

In 2009, Iwata et al. proposed Topic Tracking Model (TTM) [42], a model created for analyzing consumer purchase behavior on e-commerce websites. It borrows the idea of passing Dirichlet priors to future time periods from DTM [10], but it changes how this passing on occurs. The authors attempt to model the items that a user buys at each time period as topics, in order to inform future time periods of items likely to be bought together. In the abstract concept of topic models, users in their scheme represent documents, and the items that a user buys represent individual words in a document. Because it was designed to track the trends of consumers, they pass user-specific priors into future time periods. Whereas in DTM the topic distribution and global γ prior are passed onto the next time period, TTM passes on the topic distribution and a specific distribution for each user representing their interests, i.e. their frequently used words. This effectively allows the model to track

not only the evolution of topics throughout time, but the individual evolution of users throughout time. The authors also allow for "long-term dependencies" to be passed on, where instead of just passing the topic distribution and interests from the previous time period, those of the past L time periods are passed along. These long-term dependecies are included with the acknowledgment that users' future interests are not solely derived from the most recent time period.

The authors test TTM against multiple variations of LDA, on two data sets of transactions. The data sets contain transactions from movie and cartoon downloading websites, respectively. The former contains 70,122 users, and the latter contains 143,212 users. To show the capabilities of their model, they show the N-best accuracy of TTM and the baseline models, which measures the percentage that purchased items are contained in the set of N-highest probability items. The accuracies of the movie data set are low for every model, but TTM beats out all LDA variants on both data sets. The authors also show that TTM's runtime is much faster than the slowest LDA variant, and only 30-40% slower than the fastest LDA variants (including Banerjee and Basu's version of online LDA [6]).

This is clearly a novel implementation of a topic model in a temporal setting, and although it moves away from the traditional use of topic models on documents, it lays out a framework for working with dynamic topic models. If a document can change over time, for instance if we could call a chain of Reddit comments a single document, then this approach could be a foundation for temporal modeling in social media data.

2.4 Online Topic Models

The early implementations of LDA and its descendants were very computationally expensive. They also relied on a fixed vocabulary that was determined during ini-

tialization of the model. However, there are times when new documents are added that contain some new words. In this section, we explore how LDA and HDP were transformed from the original batch-based Bayesian inference to online versions that require fewer computations, and allow for additional vocabulary to be added after initialization. This can occur if the data set is too large to load into memory all at once, or if the data set is being continuously generated (streaming setting), such as from a streaming API or RSS feed.

2.4.1 Faster Topic Model Inference

In 2008, Porteous et al. introduced a faster implementation of a Gibbs sampler for LDA that was up to eight times faster than the original [75]. The main contribution of Porteous et al. is that instead of inferring a document's probability of being in every topic, they infer only the probability of a document being in a few of the most likely topics for that document. This saves a lot of inference time in large data sets, and the authors show that in two large data sets with large numbers of topics, the top twenty topics for a given document explain about 90% of that document. The authors tested their models on four data sets. The first contained 1,500 papers from the NIPS conference, the second contained 39,861 emails from Enron, the third 300,000 news articles from the New York Times, and the final contained 8,200,000 abstracts from the PubMed journal. The authors show that on the smaller data sets, the speedup of their model is considerable over LDA, but that the effect is diminished in the larger data sets.

In 2009, Yao et al. described multiple approaches to making the inference of LDA faster while retaining accuracy [106]. The authors introduce three different variations of Gibbs sampling that improve overall inference speed. All methods assume that inference is being performed on smaller chunks of data, adding more chunks in

each iteration. The first method resamples all topics over all documents (new and old) each iteration. This is faster than normal Gibbs sampling because it is steadily increasing the number of documents per iteration instead of inferring the entire data set every iteration. The second method resamples topics only for new documents. This is much faster than the previous method, because it fixes the topic assignments of previously seen documents. It uses those topic assignments as training data for assignments of topics to future documents. The third method is an online version of Gibbs sampling, which independently processes each document. The topic-word distribution used draws only from the current document and the training data. They also propose a logistic regression classifier and Naive Bayes classifier as other means of performing inference. The authors perform experiments on relatively small data sets (1,740 NIPS papers and 51,616 paper abstracts). Despite the small data set size, the authors demonstrate how different inference methods perform in comparison to each other. Yao and colleagues show that the online versions of their Gibbs sampler converges much faster than the traditional Gibbs samplers on these small data sets. Although their versions are faster, the authors note that each individual iteration of inference is much longer than those of the traditional Gibbs samplers.

2.4.2 Online LDA

In 2007, Banerjee and Basu [6] wrote a short survey of topic models for text streams. They implement batch and online topic models, and present the first online scheme for LDA. Their approach is straightforward, running LDA [12] on small batches of documents in a sliding window format. The topic distribution and Dirichlet priors are initialized for the next batch using the approximations from the previous batch. The authors show that their version of online LDA is much faster than the batch version. They also test their models using normalized pointwise mutual information,

the earliest occurrence that we have seen of its use in topic model evaluation. They use the full Twenty Newsgroups data set [52], which contains approximately 20,000 news articles, and five small subsets of the data set to test the models.

In 2010, Hoffman et al. developed an online variational Bayes (VB) algorithm to allow for faster inference of topics using LDA [38]. The authors created the online VB algorithm by computing an approximate expectation step based on the current observed document and the prior topic-word distribution. The authors compute the approximate distribution, assuming that the entire corpus consists of this one document. They then adjust the true topic-word distribution to a weighted average between the previous distribution and the approximated distribution. This new online version relies only on the current observed document, and the approximated topic-word distribution from all previously observed documents. In implementation, instead of observing one document at a time, the authors observe "mini-batches," or small collections of a handful of documents. This allows for more accurate, but still fast, inference of topics.

In their experiments, the authors show that increasing batch size from one up to 256 documents results in better accuracy in terms of perplexity. The authors show that on a corpus of 3.3 million Wikipedia articles, it took three days to complete, with about half of that time devoted to the inference (preprocessing accounted for the other half). They note that even one iteration of the batch version of the algorithm would have taken many days to complete. The authors do not provide specifications for the machine that experiments were performed on, so it is not possible to say how long such experiments would take today. However, for the sake of comparison between online and batch LDA, it seems safe to say that online is considerably faster.

2.4.3 Online HDP

In 2011, Wang et al. [98] developed an online variational Bayes algorithm for HDP. The authors apply the same approach, using mini-batches of documents to produce approximations of the expectation step of the expectation maximization algorithm.

Due to HDP's inherently more complex computational nature (it infers the number of topics as well), the authors used much smaller data sets in their evaluation of their online HDP variant. The authors showed results for 352,549 *Nature* articles and 82,519 *PNAS* documents. Wang and colleagues note that traditional HDP can only be run on batches of 20,000 documents due to time constraints. The authors change the experiment setup from that of online LDA, choosing to run each model for six hours, instead of running models to convergence or completion. They show that online HDP produces a higher accuracy after six hours of simulation. Like in the experiments for online LDA, Wang et al. do not provide machine specifications for their experiments. Assuming all models were run on the same caliber machine, online HDP outperforms online LDA and batch HDP significantly on the two test data sets.

While these online algorithms are an important step in the evolution of topic models, they are still limited by the number of documents they can handle. In the modern age of social media, where millions of documents are produced every hour, these online methods are unlikely to be sufficiently fast.

In the last twenty years, the documents that we input into to topic models have changed far more drastically than the topic models themselves. At the turn of the century, we input scientific articles, books, and newspaper articles. Now, we input social media posts including tweets, blog posts, Reddit posts, and other short texts. Topic modeling research has evolved to address the problems that arise when attempting to infer topics on these new types of data. There are a few new approaches, but many of

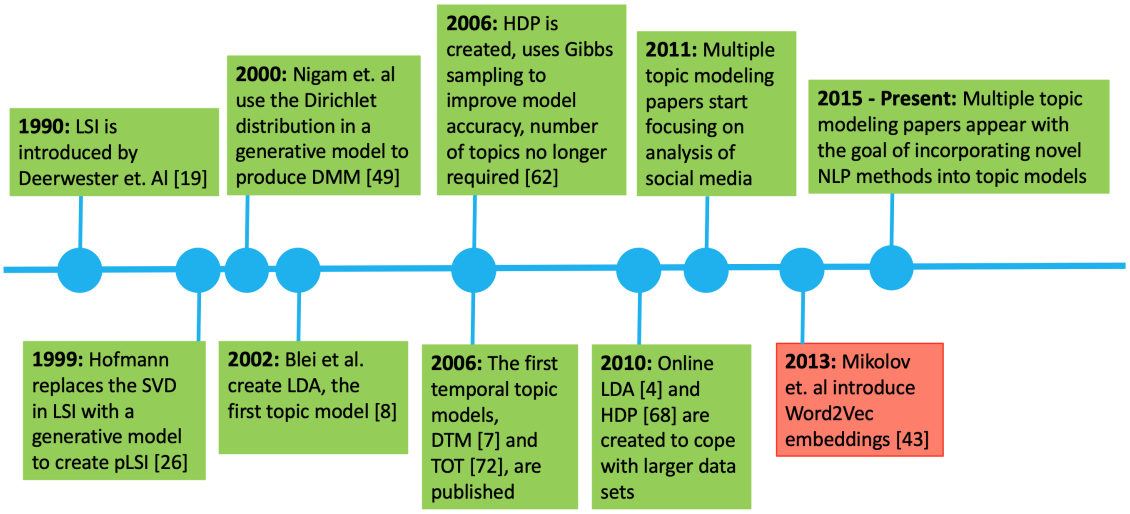


Figure 2.1: A Timeline of Topic Models from Inception to Today

the old mathematical components from twenty years ago are still the accepted best practices. In each of the following subsections, the first model is the seminal model for the category of models, and we explain it in depth, followed by descriptions of other models in the category.

2.5 Modern Topic Models

Figure 2.1 depicts the bigger developments in topic modeling's thirty year history. In this section, we cover the most recent three dots on the timeline — the application of topic models to social media data, the invention of Word2Vec, and the incorporation of natural language processing techniques into topic models.

2.5.1 Non-negative Matrix Factorization

Non-negative matrix factorization (NMF) is a mathematical process in which a matrix of non-negative values is factorized into two new matrices such that the product of the two new matrices is equal to the original. Like the inference of generative models,

NMF is known to be NP-Hard [92]. The two factor matrices have a much smaller dimensionality than the original matrix. In the case of topic modeling, the large matrix is the document-word matrix. In other words, the large matrix is the set of documents, where each document is represented as a vector of words. The two smaller matrices that NMF factors the large matrix into are the topic-word matrix and topic-document matrix. This topic-word matrix can be interpreted as a topic distribution over words, and topics can be derived from it by choosing the highest weighted words for each topic. Likewise, the topic-document matrix can be interpreted as a topic distribution over documents, and each document's individual topic distribution can be derived from this matrix. A k value representing the number of topics desired by the user decides the final dimensionality of the smaller matrices. If there are D documents and V words in the vocabulary, the large matrix will be of size DxV, and the smaller matrices will be of size kxV and Dxk, respectively. The setup of the problem is simple, but solving it exactly is intractable, so an approximation algorithm is used. NMF works by minimizing an error function between the original matrix and the product of the two factor matrices. Non-negative matrix factorization has been shown to be mathematically equivalent to pLSI (described in Section 2.2) when using KL-divergence as the error function [36].

In 2006, Shahnaz et al. demonstrated NMF's capabilities as a topic model [86]. The authors use a sparse encoding of the document-term matrix to save memory and run the algorithm more efficiently. The experiments consisted of two data sets, one with 21,578 Reuters news articles, and the other containing 7,919 transcripts from different news sources. The authors limit their experiments to documents pre-labeled with a single topic. In the first data set, the accuracy of the model ranges from 96% when they label documents using only two topics, to 54% when labeling twenty topics. The second data set ranges from 99% accuracy at two to 80% at twenty topics. The

authors attribute the disparity in performance to the broad topic labels in the Reuters data set.

In 2011, Kasiviswanathan et al. published a temporal model that used NMF as its basis [47]. While NMF was not designed specifically for a streaming setting, the authors used NMF on documents in each epoch in order to produce sets of *emerging topics*. Emerging topics are temporal topics that are receiving more attention, i.e. rising. The authors tested their algorithm on a set of 9,394 documents, and on the Twenty Newsgroups data set, consisting of 18,774 news articles. Instead of other topic models, the authors compared their model against clustering algorithms. The authors order the documents not by date but by pre-labeled topic, and introduce documents in order of their clusters with some overlap in each epoch. While the model itself is unsupervised, the ordering of the documents in their experiments constitutes some level of supervision. In real life data streaming settings, data are not generated in order of their pre-labeled topic. This potentially gives an unfair accuracy advantage to the model, so the results in this paper should not be directly compared with those of other topic models. After testing on the two pre-labeled data sets, the authors also test their model on a set of 5,199 tweets about the performance of IBM's Watson on Jeopardy. The authors ran the model on two weeks' worth of tweets leading up to the airing of the show (8,434 tweets) as a baseline for finding emerging topics.

In 2013, Yan et al. took a new perspective of the NMF approach [104]. Instead of attempting to detect topics using a document-term matrix, they employ a term correlation matrix in order to detect topics in short texts. They argue that word co-occurrence within documents is not adequate for detecting topics in short texts due to the low frequency of words within these documents, and that the true nature of topics in these texts can be captured by analyzing the correlation between words themselves. Instead of building a matrix of documents with word counts, the authors

build a matrix of words with positive pointwise mutual information scores (positive to maintain non-negative factorization). The authors employ NMF on this matrix in order to learn the topics directly from word correlations rather than from document correlations. The authors test their algorithm on three data sets consisting of 4,520 tweets, 2,630 news article titles, and 36,219 online questions, respectively. The authors compared their algorithm to LDA and other implementations of NMF for topic modeling. The authors evaluate topic sets based on purity, normalized PMI and ARI.

NMF and its variants are a dimensionality reduction approach to topic modeling. In environments where noise is pervasive, dimensionality reduction methods that are able to remove noise and extract features from sparse high dimensional spaces is a potentially important future direction given the increase in both dimensionality and sparsity of modern document collections.

2.5.2 Graph-based Models

Recently, some work has focused on finding topics using graph-based approaches. When building the graph, words are generally nodes and their co-occurrence is represented as a weighted edge. The basic intuition is that words in tightly connected communities represent topics.

An early implementation of a graph-based topic model is that of Cataldi et al. [18], a model designed to detect emerging topics. In their model, the authors employ a directed graph, where each node is a word, and each edge is a weighted co-occurrence of two words. The edges are directed because the weight can be different in each direction. The edge weighting is calculated based on the co-occurrence frequency of the word pair relative to the total frequency of the word at the source of the edge. Therefore, a word with very high frequency, but many co-occurring words, is likely

to have lower outgoing edge weights than one with a lower frequency but fewer co-occurring words.

The graph is created for each time epoch in sequence, and the frequency of words in each epoch are saved for the future epochs. Once the graph has been created, a localized edge pruning is performed, removing lower-weight edges. Due to the necessity of pruning edges, this algorithm must iterate through all edges in the graph. The number of possible edges is $\frac{|V|(|V|-1)}{2}$, where $|V|$ is the number of words in the vocabulary, so the complexity of the algorithm is $O(|V|^2)$, although due to graph sparsity, the runtime is usually much faster. Topics are detected on this graph by running a depth-first search (DFS) out from a set of "emerging terms." To compute which terms are emerging in each epoch, their frequencies from previous epochs are compared to the current, to determine if the frequency has increased significantly from the past epochs. From each term reached during the DFS, a new DFS is run. If one of these DFSs reaches the original emerging term, then the term is added to a topic containing the emerging term. This process also guarantees that no two topics will include the same term, resulting in zero topic overlap. This double-DFS approach to finding topics in a directed graph ensures that any two words included in the same topic are strongly connected to each other, at least through some chain of other strongly connected words.

Cataldi and colleagues show their results on a generic Twitter data set containing more than 3 million tweets over two weeks in 2010. While they do not compare their accuracy to other topic models, they show topics that are recovered throughout the two week period. Their topics are small (four words each), but there is no analysis that shows whether the approach captures all emerging topics, or just a handful of the biggest. This work was the first to demonstrate that non-generative topic models could be used to produce reasonable topics.

In 2016, Arruda et al. designed a static topic model, topic segmentation (TS), based on an undirected graph with nouns and verbs as nodes, connected by co-occurrence [30]. The authors describe three different ways to compute co-occurrence, one based on adjacency of words, one on paragraph co-occurrence, and the other an adapted paragraph-based approach which considers co-occurrence frequency compared to a baseline frequency. The authors employ the Louvain modularity algorithm to detect communities of connected words in their graph [13]. The complexity of the modularity algorithm is $O(n\ logn)$. Given a subset of nodes in a graph, the Louvain modularity algorithm compares the number of internal edges (between nodes in the subset) to the number of outgoing edges (between a node in the subset and a node not in the subset). A higher than expected ratio of internal edges to outgoing edges indicates a high modularity score. A high modularity score, in the approach designed by Arruda et al., indicates a probable topic represented in the graph. The authors created their own data set, consisting of 200 documents built from paragraphs extracted from Wikipedia articles revolving around pre-chosen topics: actors, cities, soccer players, animals, food, music, scientists, and sports. The authors compare their approach to clustering methods instead of topic models. They do, however, show that the paragraph co-occurrence approaches to building their graph are not as accurate as the adjacency-based approach.

In 2018, Churchill et al. [27] designed a topic model, TFM, with the goal of reducing noise levels in topics in a temporal setting, and tracking the evolution of topics. The authors employed a similar directed graph structure as Cataldi et al. [18]. For the same reasons as [18], the complexity of TFM is $O(|V|^2)$. Instead of performing a local edge pruning method, the authors include an edge only if the weight meets a global threshold. This results in fewer edges in the graph, which allows for faster modeling. The authors also introduce the notion of domain-specific *flood words*. Flood

words are words that appear in documents across a domain, and like stopwords, do not add context to the topics. For example, if we were trying to understand COVID-19 relevant topics, COVID-19 would be in every topic, making it a flood word. To account for flood words, the authors introduce an upper bound for word frequency when detecting emerging terms. The authors also introduce a third parameter to capture the individual change in frequency over time for words that may be considered emerging terms. Furthermore, while Catadi et al. [18] traversed the graph using a DFS in order to find topics, Churchill et al. [27] used a breadth-first search (BFS), and limited the distance of the search using a parameter. While this more effectively filters out noisier emerging terms from topics, more parameters need to be tuned.

The authors attempt to use the emerging topics detected at each time epoch to track topics through time. In addition to the emerging terms detected at each epoch, the topics from the previous time epoch are searched for in the current epoch's graph. If the topic can be recovered in the current epoch, it is included in its current state (new terms added, or old ones lost) in the topic set for that time epoch. In this manner, the evolution of a topic can be tracked from emergence to disappearance. In their experiments, the authors used a small synthetic data set consisting of 500 documents randomly generated to consist of seven topics, seven time periods, and varying levels of flood words. The authors also tested their model at larger scale, using a set of 280,000 tweets about the 2016 U.S. Presidential Election, and a set of 14,000 newspaper articles about the same topic. The authors use precision, recall, and SNR to evaluate their model. The authors compare their results to other state-of-the-art algorithms using ground truth labeled topics, and show that overall, their model is better at detecting topics in a temporal setting than these approaches.

In 2020, Churchill and Singh proposed another graph-based topic model, Percolation-based Topic Model (PTM) [21], for detecting topics in noisy data sets. Their approach

consisted of incrementally stripping noise away from the graph, leaving small topic kernels containing little to no noise. They then used an adaptation of the clique percolation algorithm to merge topic kernels into larger, more substantive topics. Clique percolation is a way of clustering size-k cliques in a graph. It consists of grouping two size-k cliques if they share k-1 nodes. This process is repeated for all pairs of size-k cliques. Churchill and Singh relaxed clique percolation to allow for subgraphs (topic kernels in their case) to be grouped together. In their experiments, the authors used large, domain-specific Twitter data sets consisting of more than 750,000 tweets per data set. The authors used topic diversity and topic coherence (NPMI) to evaluate the quantitative quality of their models, showing competitive scores to the best models. They also conducted a qualitative analysis of their models in comparison to state-of-the-art models in order to show the ability of their models to find more concise, less noisy topics.

Graph-based approaches add a new perspective to topic modeling, taking a different direction from the earlier generative models. They do not assume knowledge of the underlying topic distribution, so they more readily find topics of varying size. In some sense, they can be more adaptive than generative models, allowing for either a global or local topic discovery. However, if flood words are not well managed, the graph structure can be very costly to traverse.

2.5.3 Mixing Traditional Topic Models with Modern NLP Methods

In recent years, there has been a shift toward incorporating more sophisticated natural language processing techniques into topic models. In contrast to early models such as LDA, which are statistical models, these new approaches leverage prior knowledge of natural language in order to improve the coherence and accuracy of the model.

The most popular form of NLP that has been incorporated into topic models is word embeddings. A survey of word embeddings by Almeida and Xexeo [2] gives an in-depth account of their development. Here, we focus on the inception of large-scale word embeddings to provide context for the topic models that follow. In 2003, Bengio et al. described how a neural model could be used to learn a distributed representation for words [7]. They named these distributed representations word feature vectors, which have come to be known as word embeddings. In 2013, Mikolov et al. introduced a model called Word2Vec for creating word embeddings that were both fast to compute and accurate [64]. They showed how their word vectors could be used to find semantically similar words. This simplicity lends a natural hand to topic models, which are tasked with finding groups of semantically similar words that represent a topic.

Biterm Topic Model. In the same year that Mikolov and colleagues introduced improved word embeddings [64], Yan et al. released the biterm topic model (BTM) [103]. Designed for short texts such as tweets and other social media posts, the biterm topic model attempts to model the word co-occurrence patterns present in a data set instead of document-level patterns. This approach makes sense in the context of short texts because document-level patterns are harder to track in documents that contain only a small number of words. The authors extract word pairs that occur in the same document and perform inference on the word pairs instead of the documents. Biterms, as they call word pairs, are generated based on a topic-word distribution. Their generative process is dependent not on documents, but on these biterms. To approximate the biterm distribution, the authors employ a Gibbs sampling algorithm not unlike that of LDA or DMM, where each biterm is generated by some topic with some probability.

Yan and colleagues show that iterations of BTM using Gibbs sampling take significantly longer than those of LDA since the set of bitterms can be significantly larger than the set of documents. On the other hand, BTM requires less memory due to the actual size of each biterm compared to that of each document. The authors performed experiments on the Tweets2011 collection, which contains 4.2 million tweets with a vocabulary of 98,857 words. They also use a data set of questions from Baidu, a Chinese Q&A website, containing over 648,000 questions. The authors show that on both data sets, their model outperforms LDA in terms of coherence and classification accuracy.

Self-Aggregating Topic Model. In 2015, Quan et al. introduced the Self-Aggregating Topic Model (SATM), to attempt to improve topic modeling on short texts [79]. The authors identify the problem of short texts lacking adequate word co-occurrence for use in traditional topic models, and attempt to solve it by aggregating short texts into longer texts that are better suited for use in LDA. In their model, the authors employ a two step process, running LDA on the data set of short texts, and using the output topics to probabilistically generate longer pseudo-texts. A pseudo-text is an aggregation of shorter documents into a single longer document for the purposes of increasing word co-occurrence. For instance, two similar documents, 'Medical Facts: Coronavirus makes your teeth fall out,' and 'Medical Facts: Coronavirus is the number one cause of cavities and root canals,' could be combined into a pseudo-text 'Medical Facts: Coronavirus makes your teeth fall out Medical Facts: Coronavirus is the number one cause of cavities and root canals.' [1] These pseudo-texts are used to generate short text snippets that represent topics, such that each snippet is generated according to one pseudo-text. These text snippets are then output as topics.

[1] These 'facts' are *not* true.

The authors test their model on 1,740 NIPS papers and 88,120 questions from Yahoo Answers. They use a pointwise mutual information (PMI) score to evaluate the coherence of their model compared to a set of hand-labeled, gold standard, topics for each data set. The authors compare their model to LDA, DMM, and BTM, and find that their model achieves a higher purity score than the baseline models. The authors do not offer a comparison of the coherence scores between their model and the baselines.

Latent Feature LDA and Latent Feature DMM. In 2015, Nguyen et al. [70] proposed using word embedding spaces (latent features) in conjunction with traditional topic models LDA and DMM. They replace the topic-word distribution of LDA and DMM with a two-component mixture of a topic-word distribution and a latent feature component. What this means is that LF-LDA and LF-DMM retain the structure of their traditional counterparts, while adding latent feature vectors for each word in the distribution. In generating a word for a document, the word is either chosen from the drawn topic, or from the latent feature vector of the topic, in essence expanding the pool of words to be chosen by allowing words similar to the topic in the latent feature space to be added as well. Because of the added latent feature space, the authors expect their models to perform well on data sets where there is little data about topic-word distributions, like short texts.

The authors test their model on different variants of the Twenty Newsgroups data set [52] containing 18,820 documents (variants contained 1,794 and 400 documents respectively), the TagMyNews data set [95] containing 32,597 documents, and a Twitter corpus containing 2,520 documents. They use NPMI to evaluate the quality of their models using Word2Vec [64] and GloVe [74] as the latent feature spaces. They show that at least one of their models beat vanilla LDA in terms of NPMI on all data sets. The authors also compare their models using purity and find that they beat

LDA in terms of purity. Finally, the authors perform a qualitative analysis of their models, showing how their model successfully groups together topics over iterations.

Neural Variational Document Model. While not technically designed to be a topic model, Miao, Yu, and Blunsom, created the Neural Variational Document Model (NVDM) in 2016 to model documents [62]. The goal of NVDM is to create a "continuous semantic latent variable for each document." The NVDM is an unsupervised generative model that uses a neural network to perform a multinomial logistic regression on the document set, resulting in what is essentially a word embedding vector for each document. The authors tested their model against LDA and other neural document classifiers on the Twenty Newsgroups data set, and a set of Reuters newspaper articles consisting of 804,414 articles. The authors use perplexity as their only evaluation metric for topic quality and show that the perplexity of their model is lower than that of the baseline models.

lda2vec. In 2016, Moody created a model called lda2vec, with the goal of directly incorporating the word2vec model into the classic LDA model [66]. In his paper, Moody alters the word2vec model to create document vectors as well as word vectors. With these document vectors, he is able to measure the similarity between documents, and between documents and words or phrases. He then represents each topic vector as a vector in the same space as the word and document vectors, where each topic vector is calculated by summing the probability of each document belonging to that topic. The resulting topic vector can be compared to word vectors to find the most similar words to the topic.

The author shows that this approach works well on Hacker News comments, containing 66,000 short texts, as well as the Twenty Newsgroup data set, containing 11,313 documents. The author does not offer any comparison of the model to other

state-of-the-art topic models, instead opting to show example topics discovered by lda2vec on each data set.

Pseudo-document-based Topic Model (PTM). In 2016, Zuo et al. proposed the Pseudo-document-based Topic Model [112] as an improvement on the Self-Aggregating Topic Model [79]. PTM assumes that short texts are generated from longer pseudo-documents. For each short text observed, a latent pseudo-text is drawn from the distribution of pseudo-documents. A topic is then drawn from the topic distribution of the pseudo-text as opposed to the short text, and that topic is used to generate the next word in the short text. The authors' hypothesis is that by condensing many short texts into a single pseudo-document, the word co-occurrence matrix is condensed, leading to more accurate approximation of topics. The authors claim that their generative process is significantly faster than the two-step process of SATM owing to the fact that their process is a single step in which a single pseudo-document is used to generate a document. The authors also propose a variant of their model, SPTM, to account for sparsity by adding the "Spike and Slab" prior [41] to the topic distribution of pseudo documents.

The authors test their model on four data sets: a news data set consisting of 29,200 articles, a DBLP data set consisting of 55,290 research paper titles from six research areas, a questions data set consisting of 142,690 questions from a Chinese question and answer website, and a Twitter data set consisting of 182,671 tweets labeled with categories. The authors use precision, recall, and f-score to evaluate their models, as well as a PMI-based coherence score. The authors perform a short qualitative analysis of PTM on DBLP, showing how the most probable topics in a pseudo-topic can be explained by the content of the pseudo-topic. The authors show that PTM or SPTM outperform LDA, SATM, and other baseline models on all data sets except for Twitter.

Embedding-based Topic Model. In 2016, Qiang et al. introduced an embedding-based topic model (ETM), for performing topic modeling on short texts with the help of word embedding vectors [77]. Using the Word2Vec framework created by Mikolov et al. [64], the authors create a new distance metric, called the Word Mover's Distance, to measure the difference between documents given the word embedding vectors of their component words. The Word Mover's Distance (WMD), an adaptation of the Earth Mover's Distance, computes the minimum distance between each word in one document to its closest neighbor in the other document. The authors compute the WMD between documents, and then aggregate short documents into longer pseudo-texts using K-means clustering, with the WMD as the distance metric. The authors run LDA on the pseudo-texts to get topic assignments for each pseudo-text and word in the vocabulary. They create a Markov Random Field (manifested as an undirected graph) and for each pair of words with a sufficiently small WMD, they create an edge in the graph between the two word nodes, representing their shared topic assignment. Then, for each pseudo-text, there is an undirected graph consisting of the nodes referring to the words in the pseudo-text. Words are then drawn from a multinomial distribution given their WMD and pseudo-text. As a result, similar words that appear in the same pseudo-text will have a high probability of being placed in the same topic, while similar words that do not appear in the same pseudo-text will not.

The authors tested their model against multiple state-of-the-art models on two data sets, consisting of 16 million tweets and 6,974 news articles, respectively. They find that their approach improves on the coherence of other models. While the example topics that the authors display show less noise than other models, their comparison is on a small set of only four topics.

Gibbs Sampling DMM. In 2014, in an effort to more effectively model topics on short text, Yin and Wang [109] revived the Dirichlet Multinomial Mixture (DMM) created nearly fifteen years earlier by Nigam et al. [71]. Yin and Wang altered the original DMM by proposing a Gibbs sampling algorithm that improved on the scalability of DMM. DMM naturally lends itself to modeling short texts because of its assumption that every document is generated from a single topic. In short texts such as social media, there is often not enough space to reference multiple topics effectively, so a model such as LDA might end up converging to a point where it finds that most documents are generated in large part by a single topic, or erroneously finding that documents are generated by many topics, leading to noisy and inaccurate topics. Yin and Wang perform experiments on their model using 11,109 news articles from Google News, as well as 2,472 pre-labeled tweets from TREC. The authors compare their model to clustering algorithms instead of topic models. Despite the small change to the original model, this new look at an old model paved the way for other more advanced variants of DMM.

GPUDMM and GPUPDMM. In 2016, Li et al. introduced GPUDMM, another topic model that incorporated word embeddings into a classic model, although in a completely different manner than Moody [54]. GPUDMM, which stands for generalized Polya urn (GPU) Dirichlet Multinomial Mixture (DMM), attempts to bring together semantically related words into the same topic using the GPU model. In the Gibbs sampling model, otherwise known as the simple Polya urn model, when a word is sampled from the vocabulary, it is promoted by inserting an additional copy of itself back to the vocabulary. In the GPU model, when a word is sampled from the vocabulary, a copy of similar words as well as the original word are added back into the vocabulary. This results in groups of similar words all being pushed to the top of a topic together, which in turn results in topics that contain more coherent

sets of words. The similarity of words in the GPU model is decided by their word embedding distance. So, in the comparison to lda2vec, which relies completely on the word2vec embedding model to create topics, the GPUDMM model amends the traditional DMM model with better words based on the word vector similarities. In terms of the DMM part of the model, the authors borrow directly from Yin et al.'s GSDMM. To incorporate the GPU model into DMM, the authors employ a probabilistic sampling strategy in an attempt to only reinforce those words which are very similar to the sampled topic.

To account for the possibility of documents being generated by more than a single document, the authors also introduce the GPUPDMM model. This sister model of GPUDMM replaces the DMM with a Poisson-based DMM, which allows for one or more topics to generate a document. The difference between PDMM and LDA is that PDMM limits the number of topics that can generate a document based on a Poisson distribution, whereas LDA allows for all topics to contribute to generation with some probability. Aside from the difference in the number of documents allowed to generate a topic, GPUDMM and GPUPDMM are identical.

Li and colleagues show results for two different short text data sets, with 12,265 and 179,042 documents respectively. In their experiments, they show that their topics are more coherent than those generated by DMM and other state of the art models that incorporate word embeddings. This approach is a promising implementation of a topic model that employs NLP methods intelligently to improve topic coherence.

Distributed Representation-based Expansion (DREx). In 2017, Bicalho et al. proposed a framework for expanding short texts using word embeddings [8]. Given a word embedding space and a document, the document can be expanded by finding the closest ngrams in the embedding space to each ngram contained in the document. For each candidate ngram, if it is sufficiently close to the ngram in the

document, the candidate word is added to the document. This process stops when no candidate words remain, or when the document size limit is reached (60 in the paper).

The authors test their framework on seven short text data sets, including four Twitter data sets, two news data sets, and one web search snippet data set, ranging in size from 1,001 documents to 70,707 documents. They show that by expanding using DREx, documents can increase in size from single digits to between twenty and sixty words (because the limit is 60).

The authors use NPMI, precision, and recall to compare the performance of models with and without DREx, and find that DREx with GloVe [74] word embeddings performs best throughout all data sets. Testing DREx with LDA [12], LF-LDA [70], and BTM [103], the authors show that DREx improves the results across the board.

Common Semantics Topic Model. In 2018, Li et al. proposed Common Semantics Topic Model (CSTM) [56] to filter noise from topics in social media data sets. Based on DMM [71] (what the authors call mixture of unigrams model), the authors add 'common topics' to the mix, allowing a document to be generated from a single 'function topic' (a traditional topic), and from the set of common topics. The common topics are designed to capture words that are commonly used across all topics. Their inclusion in the generative process hypothetically allows for a document to be generated from the true topic it represents (the function topic), as well as some noise words from common topics. Given k topics, C are defined to be common topics, and K are function topics. For each document, a function topic z_d is sampled from K, and then for each word, a topic is drawn from the mixture of $C \cup z_d$, and the word is drawn from the chosen topic.

The authors test their models on three data sets, one consisting of 12,340 web search snippets, one consisting of 179,022 questions from a Chinese Q&A website, and

the final consisting of 2,000,000 tweets. The authors tested against baseline models including: SATM [79], BTM [103], DMM [71], and DREx+LDA [8]. The authors perform a qualitative analysis of the tweets data set, showing the identified common words grouped into the common topics. The authors show that CSTM produces more coherent topics than the baseline models using NPMI, and show the top ten words of two topics from the tweets data set. The authors show that CSTM performs competitively on classification accuracy and purity. The authors also show the effect of changing the number of common topics on coherence for each data set. The coherence of topics increases up to about five common topics, and decreases as the number of common topics increases further.

Word Embedding LDA. In 2018, Bunk and Krestel proposed a model called WELDA, a combination of word embeddings (WE) and LDA [17]. In their paper, Bunk and Krestel find that word embeddings and topic models do not have a high natural correlation in terms of which words they find to be similar. They find that in terms of judging word similarity, word embeddings are far superior to LDA. With this in mind, they set out to combine word embeddings with LDA in order to create a more coherent topic model.

The authors use a pre-trained embedding model as their base embedding space, and then perform a slightly altered version of LDA's generative algorithm to find topics. Instead of just being given the topic-word distributions for each topic, an embedded topic distribution is given as well. The embedded topic distribution is a set of words that are closest in the embedding space to the top words of a given topic. Using this embedding space, for each observed word in a document, a coin is flipped with some success probability λ. If this coin flip is successful, then the observed word is replaced in the document with the nearest neighbor to the observed word in the embedded topic distribution. The replaced word is sampled instead of the actual

observed word. This results in the top words for each topic moving closer together in the embedding space, possibly helping topics become more coherent.

For their experiments, the authors tested on the Twenty Newsgroups data set and NIPS data set, which contain 11,295 and 1,740 documents, respectively. The authors note the small size of the data sets, but mention that other attempts at word embedding topic models still struggle to process these data sets. The authors use C_V as defined by Röder et al. [81] and word intrusion as defined by Chang et al. [19].

The authors found that the word intrusion scores for WELDA were higher than LDA and slightly higher than or comparable to other baseline models that used word embeddings. The authors also found that the coherence of WELDA was much better than LDA and better than or comparable to other baseline models.

Laplacian DMM. In 2019, Li et al. adapted DMM to better suit short texts, and proposed Laplacian DMM (LapDMM) [57]. While they note that the assumption of one topic per document already suits DMM to short texts, they incorporate variational manifold regularization in order to preserve the local neighborhood structure of short texts. Manifold regularization in the context of topic modeling adds the constraint that topic representations of document pairs should be similar to each other if they are nearest neighbors in document manifolds. In order to find the nearest neighbors of documents, a graph is constructed prior to training LapDMM that measures document distances. The Laplacian matrix of the graph can then be used as a constraint on the topic assignment of documents, to ensure that documents assigned to the same topic have words in similar neighborhoods in the graph. The authors use word embeddings (specifically Word2Vec [64]) and the Word Mover's Distance [50] to compute the distance between documents. Using word embeddings instead of direct term distances allows one to compare documents with no words in common.

The authors test their model variants against DMM [109], GPU-DMM [54], BTM [103], and an aggregation model similar to SATM [79]. They use three data sets, a Trec question data set consisting of 5952 documents, a web search snippets data set consisting of 12,340 documents, and a Stack Overflow data set consisting of 20,000 question titles. Using NPMI and accuracy (the topic of each document was labeled), the authors showed that their model performed better or competitively on all data sets. They also showed that LapDMM was more accurate in classifying documents on each data set. The authors note that the construction of the document graph can be time-consuming, especially for large data sets.

CluWords and CluHTM. In 2019, Viegas et al. proposed a topic model that leveraged clusters of words and TF-IDF to generate topics [93]. The model revolves around the notion of a CluWord, which is a cluster of words given an embedding space. Words belong to the same CluWord if their cosine similarity in the embedding space is greater than some threshold α. Once CluWords for each word in the vocabulary have been computed, each word in each document is replaced by its CluWord. The TF-IDF of the CluWords are then computed to filter out very common and rare CluWords. The authors use CluWord representations of documents with NMF to produce a topic set. The authors test their model on twelve small data sets (909-22,384 documents each), against a number of baseline models including LDA [12], BTM [103], GPUDMM [54], and ETM [77], and find that their model's coherence outperforms the baselines.

In 2020, Viegas et al. proposed a model based on CluWords and NMF called CluHTM to perform hierarchical topic modeling [94]. The model is initialized in the same manner as CluWords [93], but once the initial set of topics is approximated, instead of finding CluWords and performing modeling on the entire data set, it recursively focuses only on the documents belonging to a single topic at a time. This repeated process results in a hierarchy of topics where each set of topics one rung

down corresponds to one of the original topics and so on. The authors test CluHTM on the same data sets as in the CluWords paper, this time testing against other hierarchical models. They find that in most data sets, CluHTM outperforms the other models in terms of coherence.

Embedded Topic Model and Dynamic Embedded Topic Model. In 2019, Dieng, Ruiz, and Blei introduced another version of a topic model assisted by word embeddings [35]. In their model, also named ETM, words and topics are both represented by a vector in an embedding space. Because topics and words are projected onto the same embedding space, words can be placed into topics probabilistically based on how close a word vector is to the topic vector. The generative process described is similar to that of LDA, where for each document, a topic is drawn according to a probability distribution. In the case of LDA, that distribution is the Dirichlet. In the case of ETM, that distribution is the logistic-normal distribution, in order to facilitate easier reparameterization in the inference algorithm. Then, for each word, a topic assignment is drawn. Finally, the observed word is drawn, in embedding form, from the assigned topic. In this way, the words are drawn from their context (the embeddings), instead of from the words that are close to them in the given document. The ETM model is fit using a neural network with the goal of maximizing the log marginal likelihood of observed documents.

ETM can be used with or without pretrained embeddings, and in the case of pretrained embeddings, will assign words in the embedding space to topics even if they do not occur in the corpus. This could be particularly useful in sparse data sets like domain-specific Twitter data, where there is a limited vocabulary and little word co-occurrence reinforcement.

The authors compare their model to LDA and NVDM. The authors test their model on the Twenty Newsgroup data set, and a data set of over 1.8 million news

articles from the New York Times. They use a hybrid evaluation metric, which is the normalized product of the topic coherence (normalized PMI) and topic diversity, which the authors define to be the percentage of unique words in the top 25 words of all topics. This hybrid metric represents a balance between coherence and accuracy that seems not to be achieved by any previously used metrics. The authors find that the ETM with pretrained embeddings performed best on Twenty Newsgroups, consistently outperforming other methods. On the New York Times data set, other models catch up to it, with a much closer spread in terms of interpretability and predictive power. However, it still outperforms the other models. The authors also show that ETM is robust to stopwords. If stopwords are left in the documents, it finds a few stopword topics, but stopwords do not infiltrate other topics as is the case with many other topic models.

Also in 2019, Dieng, Ruiz, and Blei designed the Dynamic Embedded Topic Model for temporal topic modeling assisted by word embeddings [34]. A variant of the ETM that they first designed, D-ETM adds a time-varying aspect to the model. The difference in the model's generative process is that the generation is run for each time step, such that there are k topics in each time step, still all projected onto an embedding space. Word embeddings are not time dependent in this model, so it is possible that topics from an earlier time step could pick up words that appear only in documents in later time steps. This helps for continuity of topics through time steps, but may be a problem when judging the accuracy of a topic at a given time.

The authors test their model on three temporal data sets, including ACL abstracts (8,936 documents), articles from *Science Magazine* (13,894 documents), and United Nations general debates (196,290 documents). They tested against two dynamic versions of LDA (D-LDA and D-LDA-REP). The authors use the same evaluation metrics, topic coherence, topic diversity, and topic quality, and find that In the UN data

set, D-ETM beats D-LDA on topic diversity and quality, but loses in terms of coherence. The topic quality difference between D-ETM and D-LDA is just 0.001. In the Science Magazine articles, results follow a similar pattern. In the smallest data set, ACL, D-ETM performs the best across all metrics. The authors also find that D-ETM beats D-LDA on two of three data sets in terms of perplexity.

Topic Modeling with BERT. In 2020, Thompson and Mimno proposed using the language model BERT [33] to produce topics. The authors use k-means to cluster tokens observed in the data set based on their contextual vectors drawn from BERT. BERT is a language model originally proposed in 2018 that is a contextual word embedding space. It is a bidirectional model, meaning that its word embeddings consider both the left and right side context [33]. This differs from previous models such as GloVe and Word2Vec which are context-free embedding spaces with a single embedding representation for each word. It excels across the board on traditional NLP tasks. The authors propose different model variants based on different variants of the BERT model. In their experiments, the authors employ three data sets, including a 1,000 document Wikipedia data set, a 5,300 document data set consisting of U.S. Supreme Court opinions, and a 25,000 document data set consisting of Amazon product reviews. The authors test their model variants against LDA using PMI-based topic coherence, and diversity (which they call exclusivity). The authors show that for any given metric, at least one of their model variants outperforms LDA; however, no model consistently outperforms LDA on every metric. In a qualitative analysis, the authors show how their models are more syntactically-aware in their clusterings than LDA.

NLP-based topic models have evolved in the past seven years, mostly due to the innovation of Word2Vec in 2013. The best of these models incorporate word embeddings in some smart way, not to wholly replace classic topic model frameworks, but to

help reduce the sparsity of the word co-occurrence space. As shown by Dieng et al. in 2019, these models continue to get more accurate, and the evaluation metrics continue to evolve, honing in on important aspects of topic models in different scenarios.

Meta-data Augmented, Supervised, and Reinforcement Learning based Models. While the scope of this survey is focused specifically on unsupervised topic models, we would be remiss if we did not mention the meta-data augmented, supervised, and reinforcement learning based topic models that have begun to appear in recent years.

In 2011, Zhao et al. proposed Twitter-LDA [110] as a means of producing better topics on Twitter data. Their model worked by aggregating tweets into larger documents by user. These pseudo-documents are then fed into an LDA-based model that assumes a different topic distribution for each user. This approach has shown promising quality on Twitter data, but it requires meta-data (namely, the user who sent each tweet). It also requires the ability to go back and recover the entire tweet set of each user in the data set, which can be impractical for larger data sets. In 2014, Sasaki et al. proposed a hybrid model of Twitter-LDA [110] and Topic Tracking Model (TTM) [42] called Twitter-TTM [84]. It essentially inserted Twitter-LDA's generative model into the TTM temporal framework to produce a temporal model that works well on Twitter data, given the meta-data and extra user tweet sets required by Twitter-LDA. The authors show that Twitter-TTM is much better on their Twitter data set than LDA, Twitter-LDA, and TTM in terms of perplexity.

Thanks to the rise of neural networks, we have started to see these new models appear in earnest. While this is not an exhaustive list, we highlight a few of the more seminal works. One supervised Hierarchical Topic Modeling approach uses supervision in the form of a labeled topic hierarchy to allow for a more accurate hierarchical topic structure [59] (this approach uses a graph structure as opposed to neural networks).

In 2019, Adversarial-neural Topic Model (ATM) [99] uses a generative adversarial network (GAN) to approximate topics through reinforcement learning. It works by translating each document into TF-IDF vectors and asking the GAN to approximate Dirichlet priors such that it can recreate the document with high accuracy. Generative Adversarial Networks are neural networks that contain a generator neural network and a discriminator neural network. The generator is tasked with creating an acceptable fake version of the document, and presented with both the true and fake document, the discriminator is tasked with guessing which is the fake document. In doing so, the authors hope that the GAN can learn which are the most important words in the data set and learn patterns in their appearance in similar documents. In 2020, another reinforcement learning algorithm, Bidirectional Adversarial Topic Model (BAT), [100] was proposed to build on ATM. Instead of just the generator network and discriminator network, a third component of the framework was added: the encoder. The encoder takes a V-dimensional document representation (where V is the size of the document), and converts it to a K-dimensional topic distribution (where K is the number of topics). Now, instead of just having to verify that the document is similar to the real document, the discriminator is tasked with evaluating the pair of the generated document *and* its simultaneously generated topic distribution. The data sets that they test on are relatively small and do not contain short texts, but their coherence results on those data sets are strong.

There have been a handful of semi-supervised topic models proposed in recent years. A few semi-supervised topic models have been built with a specific application in mind, such as predicting labels based on product reviews [55], learning visual concept classifiers from labeled images [111], and learning image labels for use in image recognition [102].

The area of semi-supervision that we focus on entails semi-supervision on a word-level. Instead of providing labels for documents, small sets of 'seed words' are labeled. In 2009, Andrzejewski et al. proposed a model that allows users to encode pairs of words that should (Must-links) and should not (Cannot-links) be placed together in a topic [3]. To accomplish this, the authors encoded the topic-word distribution as a set of Dirichlet tree distributions, called a Dirichlet Forest (DF), to produce the model DF-LDA.

DF-LDA was extended by Kobayashi et al. to allow for more complex logical expressions than Must-link and Cannot-link [48]. Expressions such as *and, or, and negation* were added to allow for easier linking of groups of words. One weakness of DF-LDA is that in order to model increasingly large numbers of constraints (Must-links, Cannot-links, and other expressions), the number of Dirichlet Forest models must be increased substantially. In order to translate a set of seed topics into constraints, dozens of Must-links must be added. This inhibits the ability of DF-LDA to scale to larger data sets.

Guided LDA (GLDA) was proposed to allow users to provide guidance to LDA in the form of 'seed words' [43]. GLDA models two separate topic-word distributions: one unsupervised and one supervised by the seed topics and then associate the seeded topics and the unseeded topics. Documents are generated as a mix of the two distributions, the latter distribution generating only seed words.

More recently, Gallagher et al. proposed Correlation Explanation (CorEx) Topic Model [37]. Similar to GLDA, the authors used 'anchor words' for topics. Unlike generative topic models, topic sets are created by grouping words by correlation to the anchor words. Corex, which is not a generative topic model, does not allow for words to be placed in multiple topics with different probabilities. This can result in inflexible

topics. Furthermore, the reliance on word correlations makes CorEx vulnerable to clusters of high frequency noise words that are more prevalent in social media.

Category-Name Guided Text Embedding Topic Model (CatE) [61] is a non-generative model that learns the embedding vector of each document in the data set as well as the vectors of each word using a word embedding space to iteratively select similar words to seed words for each topic.

Chapter 3

Notation and Evaluation Methods

In this chapter, we define the universal notation and some of the evaluation methods that we use throughout this dissertation. Certain chapters will define their own notation and evaluation methods on top of what is defined here.[1]

3.1 Notation

Let D represent a *data set* consisting of M documents or posts, where $D = \{d_0, d_1, ..., d_{M-1}\}$. A *document* d is an ordered collection of N words, where $d = \{w_0, w_1, ..., w_{N-1}\}$. Two words *co-occur* if they appear in the same document. The *co-frequency* of a pair of words is the number of documents that the pair co-occur in.

We consider words that do not add to the interpretability of topics to be *noise words*, and those that do to be *content words*. Noise words come in two different types: *context-specific* and *independent*. Context-specific noise refers to words that are noise *only in the context of the domain of the data set in which they appear*. Examples of context-specific noise words would be words like 'hillary', 'sanders', 'trump', and 'campaign' in a data set about the 2016 United States Presidential Election. Context-independent noise refers to words that are noise *regardless of the context of the domain of the data set*. Examples of context-independent noise words are stopwords and other

[1] Qualitative analyses are usually tailored to the specific topic model, so we describe them in individual chapters.

words that add little to no value to a topic, like URLs, most user tags in Twitter, and misspelled words.

A topic z consists of a set of ℓ words, $z = \{w_0, w_1, ..w_{\ell-1}\}$, where the words in z are coherent and interpretable. A topic set Z consists of k topics, where $Z = \{z_1, z_2, ..z_k\}$. A noise distribution Ω consists of a set of H words, $\Omega = \{w_1, w_2, ..w_H\}$, where the words in Ω represent noise.

An *NGram* is a set of n words that appear together in the same order, in the same document. We refer to an n-gram of size 2 or greater as a *phrase*. While we focus on words, one could easily generalize a document or topic to bigrams and/or phrases.

3.2 Evaluation Methods

In Section 2.1, we describe at a high level some of the most common evaluation methods for topic models. Here, we define the quantitative methods that we use throughout this dissertation.

3.2.1 Coherence and Diversity

To assess a model's ability to detect coherent, meaningful topics, we use normalized pointwise mutual information (NPMI) [53]. NPMI is a distance measure that captures how closely related two words are given their relative cofrequency. Many recent topic modeling papers, including that of GPUDMM [54], have employed NPMI or one of its variants to assess the coherence of their models [34, 35, 77, 79].

For a pair of tokens (x, y), we define the probability of them appearing together in a document as $P(x, y)$. We use this probability to compute the NPMI of a topic $z \in Z$ as follows:

$$NPMI(z) = \frac{\sum_{x,y \in z} \frac{\log(\frac{P(x,y)}{P(x)P(y)})}{-\log(P(x,y))}}{\binom{|z|}{2}}$$

The higher the NPMI score, the higher the mutual information between pairs of words in the topic. This indicates high topic coherence, which in turn reflects on the ability of the model to detect meaningful topics.

In addition to assessing the meaningfulness of topics, we are interested in a model's ability to find distinct topics. A model that finds the same coherent topic ten times, but does not find other topics should not be considered as effective as a model that finds many unique topics that may be slightly less coherent. We measure this using topic diversity. Topic diversity is the fraction of unique words in the top 20 words of all topics in a topic set [34]. High topic diversity indicates a model was successful in finding unique topics, while low diversity indicates a failure to discern topics from each other.

Neither topic coherence nor topic diversity are indicators of a good topic set individually. However, together they can give a better notion of whether a topic model that is producing good topics. Topic coherence measures the meaningfulness of individual topics; however it does not detect the meaningfulness of the entire topic set. When coupled with topic diversity, which measures whether or not topics are being repeated or mixed together, topic coherence becomes useful.

In practice, topic coherence scores are usually within an order of magnitude of each other on any given data set. We can create a joint metric, *topic quality*, that is the product of coherence and diversity. Topic quality makes it easy to measure coherence and diversity together against another scalar, like time.

3.2.2 Topic Recall

Sometimes, we are given ground truth topics with which we compare the approximated topics of our topic models. To measure the ability of models to produce topics similar to the ground truth, we employ *topic recall*. Topic recall is the fraction of

words from the full topic that an approximated topic recovers in the top x words. For our experiments, we used the top 50 words per topic. In order to ensure fairness and not punish models for having topics that do not fall into the final topic set, for each full topic, we count only the approximated topic with the highest recall.

3.2.3 Topic Entropy

In some cases, we are interested in understanding the amount of information present in our topic set. There are a number of different ways to measure this. A simple way is to look at the probability of every ground truth word $p(w)$ in every topic and sum the probabilities. In our case, ranking is more important than probability, where the rank of a word w in topic z is the position of w in z based on $p(w)$ given z. So, the most probable word to appear in z is rank $R = 1$, and the least probable is rank $R = |V|$. The problem with this simple ranking approach is that the penalty for having a poorly ranked word is much higher than the reward for having a highly ranked word. In reality, if a topic word is ranked as the 1000th or the 3000th word, both are bad and overwhelm the information calculation. Therefore, to adjust for that, we propose using a variant of Shannon's entropy, *topic entropy*, that uses the number of digits in the ranking R of each word as a proxy for the number of bits needed to represent the topic word: $E = \sum_1^{|z|} c \times log_{10} R[(p(w), z])$, where $R[p(w), z]$ is the rank of each ground truth topic word for each topic and c is a constant. We suggest using c to further distinguish between rankings when $|V|$ is large. A low topic entropy score indicates a set of topics that are more compressed since they contain more information in fewer bits.

Chapter 4

textPrep: Preprocessing Pipeline

It had become clear that noise was the big inhibitor to topic model performance on social media data. Our first thought was that better preprocessing could probably solve the problem on its own. In the past, Schofield et al. had showed that removing stopwords resulted in better topics, but had not examined other preprocessing methods [85]. Our hypothesis that better preprocessing would solve the problem on its own was incorrect, and through extensive testing, found that while preprocessing certainly helps topic models, it is not the sole solution. Context-specific noise, in particular, is not easily preprocessed away. In the process of understanding this, we created a toolkit including a robust preprocessing pipeline with different preprocessing methods and metrics to evaluate data quality. The rest of this Chapter presents our toolkit, and shows its flexibility and usefulness in producing highly interpretable topics [23].

4.1 textPrep: Python Preprocessing Toolkit

In our experiments, we focus on three generative topic models. The first, Latent Dirichlet Allocation (LDA), is the most well-known topic model [12]. It assumes that documents are generated from a set of topics, and approximates topics by observing the documents in a data set. The second, Dirichlet Multinomial Mixture (DMM), is similar to LDA, except it assumes that each document is generated by a single

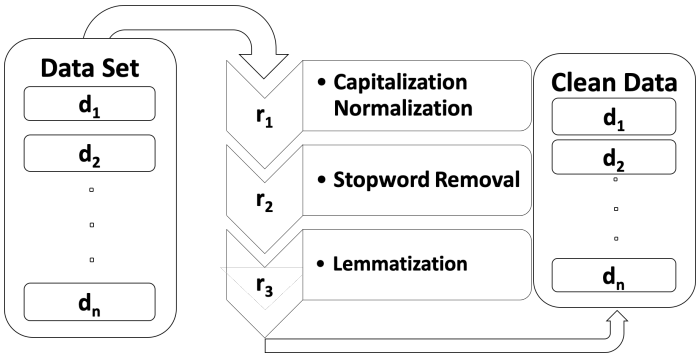


Figure 4.1: The Preprocessing Pipeline

topic [71]. Yin et al. optimized DMM for short texts [109]. The third model, Generalized Polya Urn DMM (GPUDMM) is similar to DMM, but when a word is observed, the nearest words to it in an embedding space are also sampled [54]. The sampling of related words from an embedding space should improve the coherence of topics.

To encourage more consistent preprocessing for topic models, we created a Python preprocessing toolkit for Topic Modeling (textPrep).[1] The toolkit includes each preprocessing rule we use to create our configurations, as well as a streamlined pipeline for creating other configurations. The toolkit takes advantage of other standard text processing libraries, including NLTK [9], and Gensim [80]. The rules can be easily added to configurations, or used standalone on a single document or a whole data set. Furthermore, because rule modules are designed to be used on a single document, they are ready out of the box for use in preprocessing text using PySpark pipelines. The pipeline is designed in a modular way that allows for others to add their own preprocessing rules and use them in place of or alongside the rules that are provided by default. The only requirement for a rule to be compatible with the pipeline is that a rule must be passed a document in the form of a list of strings, and return a document in the form of a list of strings. This format allows for pipelines, which also are passed and return a document in the same form, to be stacked on each other

[1] https://github.com/GU-DataLab/topic-modeling-textPrep

with as many or as few rules as one likes. The stacking of pipelines allows for one to connect multiple small pipelines in testing environments, and combine them into a larger pipeline for production environments or final stage experiments.

In addition to the preprocessing pipeline, textPrep includes data quality metrics that we use to evaluate data in this dissertation. They include vocabulary size (unique tokens), total tokens, token frequency, average token frequency, average document length, average stopwords per document, and token cofrequency. These data quality metrics are useful when deciding on the preprocessing configuration for a given data set or experiment. It is important to balance data quality metrics. High average token frequency only matters if there is still a considerably high vocabulary size – we do not want a data set with one or few very frequent unique tokens, nor do we want a data set with a million very infrequent unique tokens. Attaining better data quality can save time and resources before ever running topic models.

4.2 Preprocessing Classes and Rules

A *preprocessing rule*, r_i, is an operation that changes or removes a token, d_i. We apply a *configuration* of rules $C_k = r_1, r_2, \ldots, r_k$ to the set of n tokens in D_i to generate D_i'. As an example, a punctuation removal rule would return a document with all punctuation characters removed from each token. When the rules in C are applied to each document in $\mathcal{D}$, the result is a modified document collection $\mathcal{D}'$ (see Figure 4.1). A *topic model* $\mathcal{M}$ generates a set of m topics $Z = \{z_j | 1 < j < m\}$ that represent the themes present in $\mathcal{D}$. Each document, D_i', will be used by $\mathcal{M}$ to generate topics Z.

In our case studies, we will use different preprocessing configurations to show the importance of good preprocessing when performing topic modeling on social media data.

Table 4.1: Preprocessing Rule Descriptions

Rule	Rule Description
Elementary Pattern-based Preprocessing	
URL Removal	Removal of tokens containing URLs
Capitalization Normalization	Making all tokens lowercase or uppercase
Punctuation Removal	Removing all tokens that are punctuation
Hashtag Removal	Removing tokens beginning with the hashtag (#) sign
User Removal	Removing tokens beginning with the @ sign
Malformed Word Removal	Removing tokens accidentally created because of other rules
N-Gram Creation	N tokens are joined together to form a new token
Dictionary-based Preprocessing	
Stopword Removal	Removal of common words that do not add content value
Emoji Removal	Removing emoji and emoticon tokens
Synonym Matching	Replacing tokens that match a synonym in a given dictionary
Whitelist Cleaning	Retaining only tokens that appear on a pre-created list
Natural Language Preprocessing	
Lemmatization	Shortening a token down to its lemma using NLP
Stemming	Shortening a token down to its base by removing suffixes
Part of Speech (POS) Removal	Removing tokens that are tagged as a certain part of speech
Statistical Preprocessing	
Collection Term Frequency Cleaning	Removing tokens with a low frequency count in the data set
TF-IDF Cleaning	Removing tokens with a low TF-IDF score

We divide sixteen different preprocessing rules into four different preprocessing classes: elementary pattern-based preprocessing, dictionary-based preprocessing, natural language preprocessing, and statistical preprocessing. Table 4.1 presents the preprocessing rules (Rule), and an explanation of each rule (Rule Description). Table 4.2 gives a simple example of how each preprocessing rule works.

Elementary pattern-based preprocessing focuses on reducing the number of tokens (e.g. punctuation removal) and the variation (e.g. capitalization normalization) in tokens by searching for known patterns in tokens that may indicate noise in the context of topic identification. It also includes rules that join existing tokens to improve the semantic quality of the token (e.g. n-gram creation). Typically, these rules are implemented using regular expressions. Two rules - the hashtag removal and the user removal rules - within the elementary pattern-based preprocessing category are spe-

Table 4.2: Preprocessing Rule Examples

Rule	Example Tokens
Elementary Pattern-based Preprocessing	
URL Removal	Removed Token: aaa.com/index.html
Capitalization Normalization	Original Token: Tree Final Token: tree
Punctuation Removal	Removed Token: !
Hashtag Removal	Removed Token: #ilovechocolate
User Removal	Removed Token: @hillary
Malformed Word Removal	Removed Token: http
N-Gram Creation	Original: i love cats Bigrams: {i love}, {love cats}
Dictionary-based Preprocessing	
Stopword Removal	Removed Token: the, is, am
Emoji Removal	Removed Token: :)
Synonym Matching	Synonym: obama=barack obama= barack
Whitelist Cleaning	Only keep tokens on a pre-defined list
Natural Language Preprocessing	
Lemmatization	Original Token: better Final Token: good
Stemming	Original Token: giving Final Token: giv
Part of Speech (POS) Removal	Remove all adjectives
Statistical Preprocessing	
Collection Term Frequency Cleaning	Remove tokens that appear less than α times
TF-IDF Cleaning	Remove tokens with a TF-IDF score below β

cific to Twitter data sets since both have special meanings on that platform. The rules in this category tend to be the basic, standard ones that researchers generally apply.

Dictionary-based preprocessing focuses on using a predefined dictionary (typically manually created) to remove tokens (e.g. stopword removal), standardize tokens (e.g. synonym matching), or maintain tokens (whitelist cleaning). Typically, these rules are looking for tokens that match words/phrases in their dictionaries.

Natural language preprocessing leverages NLP techniques to normalize or remove language tokens. These techniques tend to reduce the size of the token space by understanding the linguistic similarities and differences of tokens in each document. Within this class of preprocessing, we consider three rules: lemmatization, stemming, and part of speech (POS) removal. Lemmatization identifies linguistic roots of tokens

and translates each token to that root. In order to accomplish this, algorithms consider the context and POS of the token. Stemming considers only a single token at a time (ignoring its context) and removes inflections to obtain the root form of the token. Finally, POS removal uses NLP to narrow down our vocabulary to certain parts of speech, e.g. only maintaining nouns.

Statistical preprocessing computes statistics about tokens using information about the entire collection to determine tokens that should be maintained or removed. In this class of preprocessing, we consider two rules: collection term frequency cleaning and TF-IDF cleaning. Collection term frequency cleaning refers to removing terms that have a particularly low frequency in a data set (minimum DF), or a particularly high one (maximum DF). TF-IDF (Term-Frequency, Inverse Document Frequency) looks at term relevance in a collection and removes tokens with a low relevance.

Defining this preprocessing taxonomy provides us with a second way to interpret the collection of rules that are most and least beneficial for topic modeling preprocessing.

4.3 textPrep Empirical Evaluation

This section presents case studies that use our preprocessing toolkit to assess the value of preprocessing for topic modeling. We consider three different types of social media data: Twitter, Reddit, and Hacker News data.

4.3.1 Data Sets

For our empirical evaluation, we analyze three data sets from different social media sites: Twitter, Reddit, and Hacker News. By using three different platforms, we can

Table 4.3: Data Set Properties

Data Set	# Docs	# Tokens	Tokens/Doc	Stopwords/Doc	Unique Tokens	Token Frequency
Hacker News	1,165,421	80,604,631	69.16	28.95	1,613,253	49.96
Reddit	1,022,481	28,947,427	28.31	11.54	296,132	97.75
Twitter	565,182	12,826,812	22.70	6.46	558,189	22.98

better understand how similar preprocessing rules are across social media sources. This analysis focuses on English post content.

The first data set, collected from Twitter between October 2020 and February 2021, contains tweets about the Covid-19 pandemic. Tweets were collected using the Twitter API, using over 30 unique hashtags referring to Covid-19. This data set contains over 500,000 tweets.

The second data set, collected from Reddit, contains posts about the 2020 United States Presidential Election. This data set was collected using the pushshift.io library [76]. Reddit posts were collected from subreddits related to U.S. politics and the election from September to election day on the first week of November 2020. This data set has over 1 million posts.

The third data set contains comments collected from the Hacker News platform [66]. Hacker News is a technology and entrepreneurship news site that allows users to comment and discuss articles. Collected by Moody for testing lda2vec [66], comments were only collected if the article had more than ten comments, and if the commenter had made more than ten comments in total. Overall, there are over 1.1 articles and comments.

The properties of each data set are shown in Table 8.1.

4.3.2 Preprocessing Configurations and Baselines

To demonstrate the functionality of textPrep, we create a set of preprocessing configurations and compare the similarities and differences in the resultant vocabulary set.

We choose a lightweight and heavyweight configuration for each data set, and compare each to two baseline preprocessing configurations. The first baseline consists only of tokenization and punctuation removal. This is the bare minimum that one can do to prepare data for topic modeling. The second baseline is a common set of rules used to prepare data for topic modeling, as used in the python library SciKit Learn [16]. It entails tokenization, punctuation removal, capitalization normalization, stopword removal, and the removal of tokens that appear in less than five documents.[2] Our lightweight configuration consists of the following rules: 1. URL removal, 2. Punctuation removal, 3. Capitalization normalization, 4. Stopword removal.

The difference between the lightweight configuration and the second baseline is that we drop frequency thresholding and introduce of URL removal. The heavyweight configuration consists of all of the rules in the lightweight configuration, plus: 1. Short word removal, 2. Lemmatization, 3. N-Gram Creation.

For N-Gram Creation, we used a minimum frequency of 512 for n-grams to replace their component words. To choose the threshold for n-gram creation, we tested values ranging from 64 to 1024 (powers of 2), and found that 512 offered the best balance between speed and number of n-grams created. If the threshold is set too low, it will take too long to create n-grams and too many n-grams will be created. If the threshold is set too high, few or no n-grams will be created. Both configurations for the Twitter data set, which we consider to be a special case, also include hashtag removal.

[2]The minimum number of documents varies by author, but usually lies between 2 and 10.

4.3.3 Evaluation Methods

Given that the goal of our experiments is to evaluate the effects of preprocessing on topic models and on data quality, we separate our evaluation into intrinsic and extrinsic methods. The intrinsic evaluations are meant to directly assess data quality, and the extrinsic evaluation entails evaluating the effects of preprocessing through the downstream task of topic modeling.

Intrinsic Evaluation Methods. Our primary approach for evaluating data quality is by counting the number of tokens that are removed by preprocessing. Comparing the size of the vocabulary before and after applying certain preprocessing rules can show the relative impact that these rules have on data quality. Since all configurations except for the first baseline remove stopwords, the number of stopwords per document drops to zero for those configurations. We calculate the average frequency of individual tokens before and after certain preprocessing steps. A higher average frequency of tokens indicates that preprocessing rules are successfully removing noisy or less frequent tokens without having to use a minimum frequency threshold such as in baseline 2. Furthermore, smaller vocabularies coupled with higher average token frequency make for better topic modeling conditions. A smaller vocabulary (filled with good content words) means that less memory is required to train a topic model. Topic models also have fewer words to choose from, meaning that they will be less likely to make mistakes. A higher average token frequency means that relationships between words can be more easily reinforced in the topic-word distribution, leading to more accurate and coherent topics.

Table 4.4: Data Quality Statistics for each Preprocessing Configuration on the Reddit Data Set

Configuration	# Tokens	Unique Tokens	Avg. Freq.
Baseline 1	28,947,427	296,132	97.75
Baseline 2	15,267,929	246,307	61.99
Lightweight	14,053,743	**51,399**	**273.42**
Heavyweight	**11,776,937**	326,874	36.02

4.3.4 Topic Modeling Algorithms

In order to judge the effects of preprocessing on topic models, we test against multiple models. We use LDA [12],[3] DMM [109], and GPUDMM [54].[4] These three topic models represent different approaches to generative topic modeling, as discussed in Section 2.

4.3.5 Reddit Case Study

Table 4.4 shows the data quality statistics for each configuration on the Reddit data set. The 'Avg. Freq.' column shows the average frequency of a token in the data set after being preprocessed using the configuration. The Reddit data quality shows that the configuration that garners the smallest vocabulary is the lightweight configuration. This seems counter-intuitive at first, because the heavyweight configuration includes the entire lightweight configuration, and the heavyweight configuration has over two million fewer tokens in total than the lightweight configuration. The difference is N-Gram creation. N-Gram creation is one of the few preprocessing rules that adds unique tokens instead of removing them. While it may reduce the total number of tokens even further by combining multiple tokens into one, this process also adds a unique token for each n-gram that it creates. N-grams can help identify very valuable content, but they are not always a good thing. Given that most topic models excel

[3]Specifically the Mallet implementation of LDA [60]

[4]The implementations from the Short Text Topic Model survey [78]

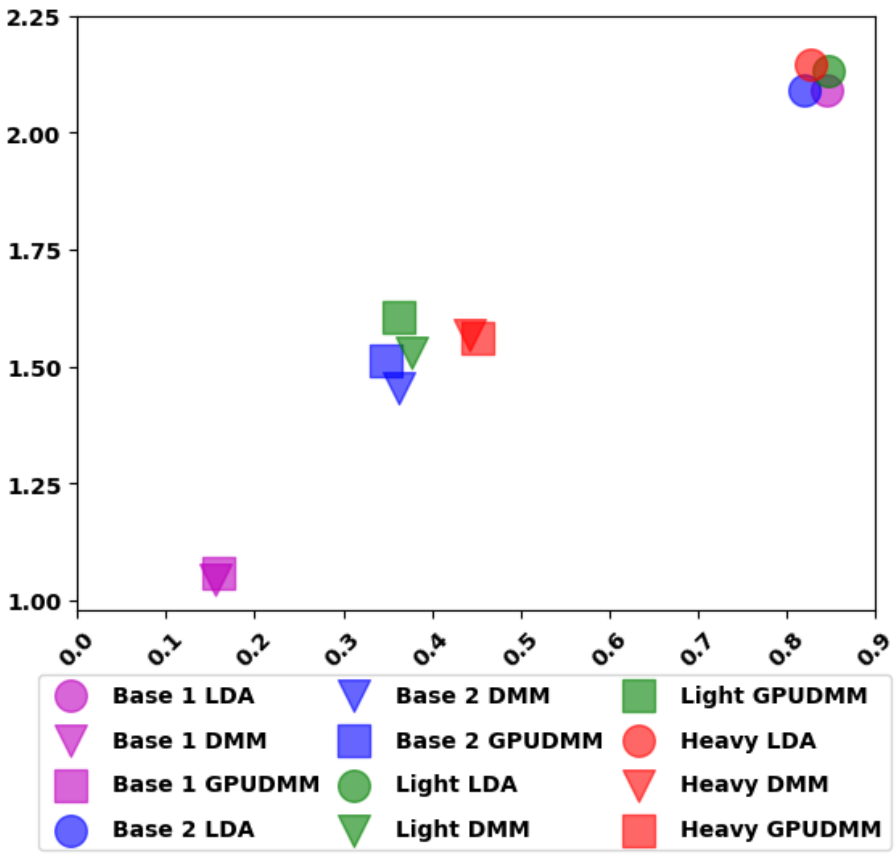


Figure 4.2: Topic Coherence (y) and Diversity (x) Scores on the Reddit Data Set

when there is a smaller vocabulary and word co-occurrences are less sparse, creating so many n-grams may negatively impact topic model performance in the end.

In order to find out if the heavyweight configuration is negatively effected by the introduction of so many n-grams, we turn to topic quality metrics. Figure 4.2 shows the performance of each model on each configuration. Topic coherence is plotted on the y-axis, and topic diversity is plotted on the x-axis. LDA is represented by circles, DMM is represented by triangles, and GPUDMM is represented by squares. Baseline 1 is colored purple, baseline 2 is blue, lightweight is green, and heavyweight is red. We see that the heavyweight configuration does not necessarily give worse topics than the lightweight configuration. In LDA and DMM, heavyweight gets a slightly

Table 4.5: Data Quality Statistics for each Preprocessing Configuration on the Hacker News Data Set

Configuration	# Tokens	Unique Tokens	Avg. Freq.
Baseline 1	80,604,631	1,613,253	49.96
Baseline 2	39,943,951	135,754	294.24
Lightweight	39,991,122	**134,609**	**297.09**
Heavyweight	**34,374,771**	1,196,609	28.72

higher topic coherence than lightweight and baseline 2. In GPUDMM, lightweight wins on coherence, but gets a lower diversity than heavyweight. Another important point that we see in Figure 4.2 is that preprocessing can effect models differently. In LDA, which earns the best coherence and diversity scores, baseline 1 is competitive with the rest of the configurations. However, in DMM and GPUDMM, using more thorough preprocessing configurations is important, and lifts coherence by 45-57%, and diversity by 4-30%.

4.3.6 Hacker News Case Study

Table 4.5 shows the data quality statistics for each configuration on the Hacker News data set. Hacker News, despite having only about 140,000 more documents than the Reddit data set, has over 80 million tokens, making documents over 69 tokens on average. This difference in initial data size leads to different results in data quality after configurations are applied. The heavyweight configuration can only reduce the total number of tokens to 34 million, still over five million more tokens than the unprocessed Reddit data set. Second, the lightweight configuration fails to significantly lower the number of unique tokens compared to baseline 2. In fact, the two configurations lead to nearly identical data quality statistics.

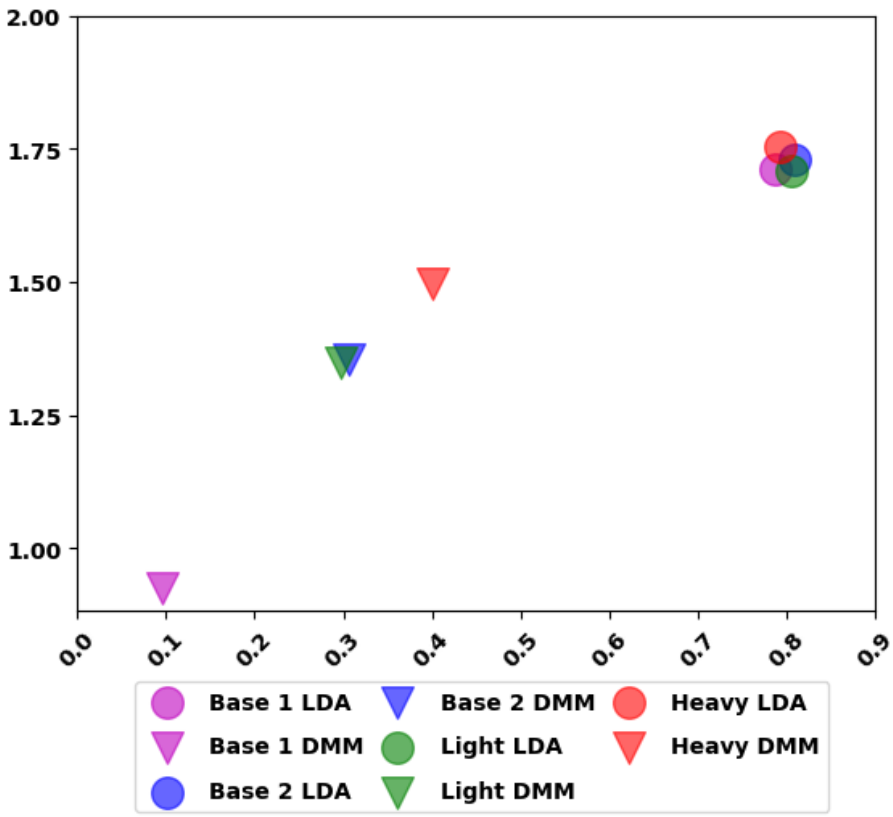


Figure 4.3: Topic Coherence (y) and Diversity (x) Scores on the Hacker News Data Set

Figure 4.3 shows the topic coherence and diversity scores for LDA and DMM on each preprocessing configuration. GPUDMM is not shown because, as a word embedding aided model that relies heavily on memory, it failed to complete on a server with 77GB of memory due to the size of the Hacker News data set. Focusing in on the baseline 2 and lightweight configurations, Figure 4.3 shows us that the lightweight configuration edges out baseline 2 for LDA and DMM, while the heavyweight configuration performs the best overall. The reversal of the lightweight and baseline 2 configurations on the Hacker News data set is another mark of how different it is from the Reddit and Twitter data sets. As we saw in the Reddit data set, and as we

Table 4.6: Data Quality Statistics for each Preprocessing Configuration on the Twitter Data Set

Configuration	# Tokens	Unique Tokens	Avg. Freq.
Baseline 1	12,826,812	558,189	22.98
Baseline 2	7,983,422	505,170	15.80
Lightweight	7,270,421	**62,879**	**115.63**
Heavyweight	**6,323,070**	337,741	18.72

will see in the Twitter data set, the lightweight configuration beats out baseline 2 on these noisier data sets that consist of smaller documents and more URLs.

4.3.7 Twitter Case Study

Table 4.6 shows the data quality statistics for each configuration on the Twitter data set. Similarly to the Reddit data quality, we see that the lightweight configuration produces a far smaller vocabulary and a far higher average token frequency than any other configuration. Again, the heavyweight configuration produces the least number of total tokens, but a large vocabulary (although not as large as in Hacker News). Figure 4.4 shows the topic coherence and diversity scores for each topic model and configuration combination. We see similar results to those of Reddit, indicating that they have similar qualities relative to Hacker News. However, in the Twitter data set, there is a clear benefit to thorough preprocessing for every model including LDA. Every configuration improves over baseline 1 in terms of coherence for LDA.

In order to show the flexibility of the preprocessing pipeline, we delved deeper into configurations for the Twitter data set. What if we could reduce the size of the vocabulary to a number similar to that of the lightweight configuration, or even more? In order to do this, we add to the heavyweight configuration a TF-IDF rule. The preprocessing pipeline's stacking ability allows us to stack another pipeline containing

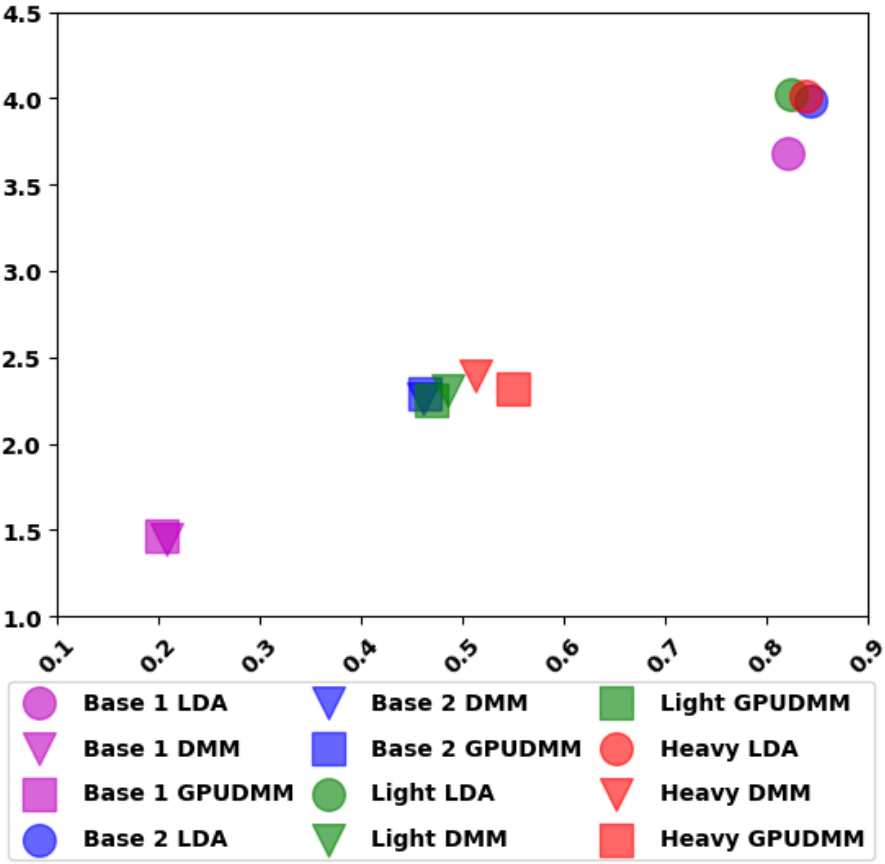


Figure 4.4: Topic Coherence (y) and Diversity (x) Scores on the Twitter Data Set

a TF-IDF rule without having to put the data through the rest of the pipeline, so we can quickly iterate through different thresholds for TF-IDF to get the data qualities that we desire. We tried using a TF-IDF rule with a threshold of 10, 1, 0.5, and 0.25. Table 4.7 shows the data quality metrics of each of these configurations compared to the original heavyweight configuration. The thresholds of 10, 1, and 0.5 were all too high, and produced very small vocabularies compared to the lightweight configuration. However, the threshold of 0.25 produced a vocabulary within about 2,000 of the lightweight. The total number of tokens is similar for thresholds between 1 and 0.25, so the real difference in data quality exists between threshold 10 and the

Table 4.7: Data Quality Statistics for TF-IDF Configurations on the Twitter Data Set

Configuration	# Tokens	Unique Tokens	Avg. Freq.
Heavyweight	6,323,070	337,741	18.72
TF-IDF 10	**5,143,060**	**7,292**	**705.30**
TF-IDF 1	5,756,844	29,101	197.82
TF-IDF 0.5	5,885,067	46,091	127.68
TF-IDF 0.25	5,968001	64,950	91.88

rest. We can see that while the threshold of 0.25 produces a similar size vocabulary to the lightweight configuration, its average token frequency is about 20% lower. With the preprocessing pipeline, we were able to quickly tailor the data qualities to our desired levels, allowing us to get to topic modeling faster. Figure 4.5 shows the results when using the TF-IDF thresholds of 10 and 0.25, compared to the lightweight and heavyweight configurations. In the case of LDA, the TF-IDF threshold of 10 produced better coherence and diversity than the rest of the configurations. In fact, only DMM sees a better topic coherence for the threshold of 0.25. Every model benefits most in terms of diversity when the threshold is set to 10. In this case, having the lowest number of total tokens, smallest vocabulary, and highest average token frequency resulted in the best topics.

4.4 Preprocessing Discussion and Best Practices

After seeing the effects of preprocessing on three unique social media data sets, it's safe to say that preprocessing is necessary, but what is the best configuration? Due to the vast differences in social media platforms in terms of data quality, we do not believe that there is truly one best configuration. Data sets can be preprocessed with a set of safe preprocessing rules, but there might be a better configuration out there that offers some significant improvements in model performance. As we saw in the

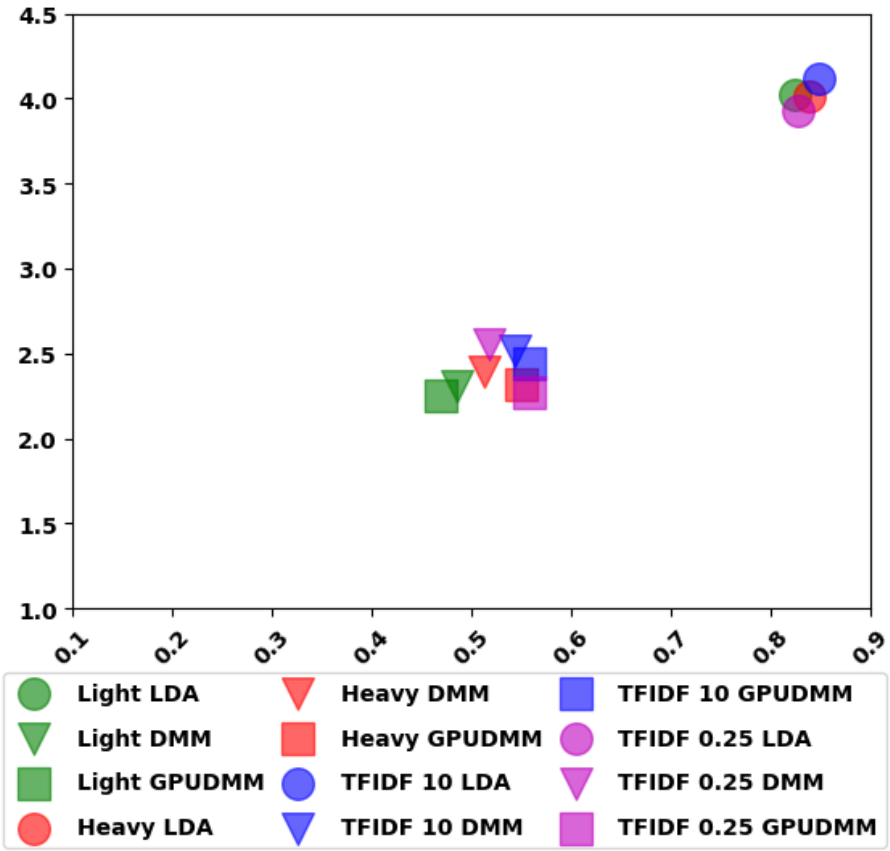


Figure 4.5: Topic Coherence (y) and Diversity (x) Scores on Twitter Data Set, using TF-IDF Rule

Twitter Case Study, the best configuration might not be one of a few likely choices. In comparing the Hacker News data set to the Reddit and Twitter data sets, we found that what's best for one data set is not necessarily the best for the next data set.

With the textPrep preprocessing pipeline, it is much easier to quickly iterate through preprocessing configurations, assess data quality, and produce better topics. To begin the process of finding a good preprocessing configuration, we recommend starting with a configuration similar to the lightweight configuration and stacking or removing one rule at a time until the data quality and vocabulary seems reasonable. The ability to filter by token frequency as in baseline 2 is built into the pipeline, as well as all of the rules that we used in these experiments. textPrep also allows for

easy integration of new rules as new social media platforms with new types of post content emerge.

Chapter 5

Percolation-based Topic Model

Once we got back on track after exploring preprocessing, we looked back to Topic Flow Model and identified some of its biggest problems. The biggest problem in our opinion was that topics are generated given only a single emerging term. We changed tacks and simplified the problem in order to focus in on topic detection without noise. We temporarily removed the temporal and online goals from our problem in order to focus solely on noise removal. We still liked the idea of a graph model, because of its ability to filter out noise and identify communities of words in data sets with sparsely-connected word co-occurrence graphs (exactly what we see in Twitter).

The state-of-the-art graph-based topic model, Topic Segmentation (TS), works by creating a graph where nodes are words, and edges are weighted by word co-occurrences [30]. TS then finds communities of words within the graph using the Louvain modularity algorithm [13]. Another state-of-the-art topic model, Biterm Topic Model (BTM), is a generative model, like LDA, that uses bigrams instead of unigrams to find topics [103].

In 2020, we proposed Percolation-based Topic Model (PTM) [21], a graph-based topic model designed to effectively filter context-specific noise from topics by leveraging ngrams to strip noise out of the word co-occurrence graph before even making topics. The remainder of this chapter describes the percolation-based topic models.

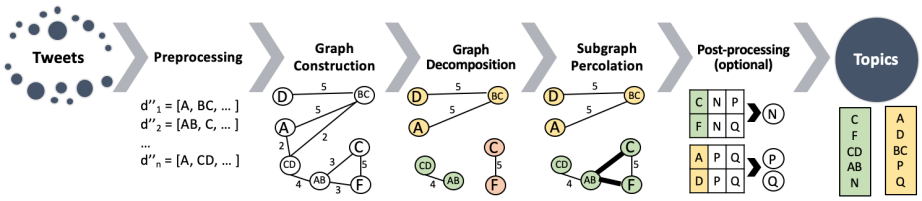


Figure 5.1: Topic Percolation Algorithm Methodology

5.1 Percolation-based Topic Model Approach

Percolation-based Topic Model is based on an adaptation of the clique percolation algorithm, which we call λ-CLIQ . The λ-CLIQ algorithm, unlike most state-of-the-art topic models, does not immediately begin generating topics as it iterates through documents. Instead, λ-CLIQ attempts to remove noise before generating topics.

5.1.1 Notation

A *token* is a word or phrase that has been preprocessed and is ready to be input into a model. We will model the document collection as an edge-weighted graph. We represent an edge-weighted graph as a pair (G, w), where $G = (V, E)$ is a graph and $w : E \to \mathbb{Z}$ is a weight function. V is the set of nodes in G, and E is the set of undirected edges in G.

Figure 5.1 shows the overall methodology. Conceptually, the core components of the algorithm can be divided into four subparts:

- Preprocessing - Tokenizing the text, where tokens are words or ngrams (phrases)
- Graph Construction - Creating a weighted co-frequency graph G containing phrase tokens and word tokens that co-occur with ngram tokens.
- Graph Decomposition - Decomposing the graph in rounds based on edge weight in order to find small, well connected quasi-cliques that can be viewed as topic kernels.

- Subgraph Percolation - Adding back edges between topic kernels to build up (percolate) larger communities of quasi-cliques by maintaining λ levels of connectivity among topic kernels, where λ represents the threshold for a graph property or metric.

Because we optimize for topic coherence, our algorithm is very selective. To increase the coverage of topics, we also propose an extended version of the algorithm, λ-CLIQ$^+$. It includes a step that uses a pre-computed domain specific language model to augment topics by adding words that are probabilistically similar to those in each topic (Post-processing).

While the overall strategy we propose mirrors that of the k-clique percolation algorithm, we do not identify full cliques, but rather well connected subgraphs. This relaxation is important in sparse domains because there are very few full cliques that occur regularly.

The high level algorithm is presented in Algorithm 1. The inputs are the database (D), a minimum frequency for an ngram to appear in the co-frequency graph G (min_freq), the increment during graph decomposition (τ), and the minimum threshold for percolation (λ). The output is the set of topics (Z). The remainder of this section describes the components of the algorithm and variants that improve the topic quality.

5.1.2 Preprocessing

The algorithm begins by preprocessing the data D to extract tokens, i.e, words and ngrams (line 4). D' is the preprocessed, tokenized data. We then identify the frequent ngrams P (these are typically phrases) (line 5). They will be a core component of the cofreqency graph (line 5). Finally, for each pre-processed tweet in D', we replace relevant word tokens with the corresponding frequent ngram token to create D'' (line

Algorithm 1 λ-CLIQ

INPUT: D, min_freq, λ, τ
OUTPUT: Z
$Z = \{\}$
$D' = \text{preprocess}(D)$
$P = \text{compute\_freq\_phrases}(D', min\_freq)$
$D'' = \text{replace\_words}(P)$
$G = \text{construct\_cofreq\_graph}(D'')$
$m = 1$
while $|E| > 0$ **do**
    $G = \text{decompose\_graph}(G, m)$
    $tf = \text{remove\_topic\_fragments}(G)$
    $Z = Z \cup tf$
    $m = m + \tau$
end while
$Z = \text{percolate\_topics(Z, } \lambda)$
$Z = \text{remove\_small\_topics}(Z))$
Return Z

6). In this way, we give priority to ngram tokens and remove the redundancy of maintaining both the individual words and the ngram in a post. For example, suppose two of the word tokens are `cat` and `dog` and that `cat_dog` is a frequent ngram. Then the *replace_ word()* method replaces adjacent tokens `cat` and `dog` with the token `cat_dog`.

5.1.3 Graph Construction

In noisy environments, we hypothesize that identifying content-rich topics requires us to identify content-rich phrases, or ngrams that appear with regularity. We accomplish this by constructing a graph that contains phrases appearing regularly in the tweet collection and words that appear regularly with those phrases.

Given a document collection D, we will model it as an edge-weighted graph. We represent an edge-weighted graph as a pair (G, w), where $G = (V, E)$ is a graph and $w : E \to \mathbb{Z}$ is a weight function. V is the set of nodes in G, and E is the set of

undirected edges in G. In our graph G, nodes represent phrase or word tokens, edges exist between two phrase tokens or a phrase token and a word token that appear in the same document, and $w(e_{ij})$, the weight on edge e_{ij}, is the number of documents containing both tokens v_i and v_j. G does not contain edges between word tokens as their frequency of co-occurrence overwhelms that of the edges between words and phrases, reducing the importance of phrases in G. Also, since many words appear together by chance on social media, focusing on phrases that occur regularly reduces some of the more 'random' connections that may appear if we have edges between nodes representing words.

Once we identify phrases P that occur at least min_freq times (line 6), the graph construction (line 7) proceeds as follows. A node is created for each phrase in P and each unigram in D'' that occurs with one or more phrases in P. Edges are added between phrases that co-occur and phrases and words that co-occur. The weight of each edge is the frequency of co-occurrence.

5.1.4 Graph Decomposition Strategies

Our model attempts to break the chains between content and noise words that negatively affect the performance of other topic models, while leaving intact communities of content words. The edges in our graph are weighted with the co-frequency of their respective vertices. The basis of our decomposition strategy is to remove edges in rounds, where all edges of a particular weight m are removed during a single round. In round one, all the edges of weight $m = 1$ are removed. Once the edges are removed, new subgraphs may emerge. Each subgraph or topic fragment (tf) is considered a *topic kernel*, removed from G, and set aside (added to Z) until the percolation phase. We continue with new rounds, increasing m by τ until no edges remain (lines 8 – 14).

Our general strategy is based on the K-Clique Percolation algorithm as described in [32]. The main difference is that we relax the clique requirement by identifying *quasi-cliques* (as opposed to cliques) that disconnect from the giant component during edge removal. Because these types of phrase graphs are sparse, full cliques are too restrictive a constraint.

5.1.5 Graph Percolation Strategies

Given our set of topic kernels $Z = tf_1 \dots tf_q$, where q is the number of topic kernels identified during the graph decomposition phase, we want to combine related topic kernels into single topics that are more coherent than the original topic fragments (line 15).

When combining topic kernels, we consider different graph properties or metrics that are possible indicators of high topic cohesion, i.e, high subgraph connectivity. One possible example graph property is subgraph density. We define λ to be the minimum threshold required in order to be a reasonably connected subgraph. Multiple topic kernels can be combined into one topic if the subgraph density of the union is greater than or equal to the subgraph density of the larger topic kernel.

A metric that is a better indicator of cohesion is Silhouette score. Silhouette score is used to determine the quality of a clustering algorithm. It measures how far apart each cluster is from its closest neighbor. We can adapt the silhouette score for our model's coherence metric. We change the silhouette score slightly to account for our closeness measure, cofrequency. We define $P(x, y)$ of two tokens x and y to be the probability that x and y appear in a document together. Instead of using minimum average distance, we use the maximum average co-frequency of a point with the points in each of the other clusters. So for a token x in topic fragment tf_i, let

$$a(x) = \frac{1}{|tf_i| - 1} \sum_{y \in tf_i, x \neq y} P(x, y) \tag{5.1}$$

To measure the distance to the closest topic, let

$$b(x) = \max_{j \neq i} \frac{1}{|tf_j|} \sum_{y \in tf_j, x \neq y} P(x, y) \tag{5.2}$$

Using $a(x)$ and $b(x)$, we can define the silhouette score of token x to be:

$$s(x) = \frac{a(x) - b(x)}{\max\left(a(x), b(x)\right)},\ if\ |tf_i| > 1,\ 0\ if\ |tf_i| = 1 \tag{5.3}$$

From Equation 5.3, we see that the silhouette score is in the range $[-1, 1]$. Higher scores indicate well-clustered, coherent topics, while lower scores indicate poorly clustered topics.

If $\lambda = 0$, we remove all topics with negative silhouette scores in order to keep as much noise out of the merging process as possible. We use a greedy algorithm to combine topics only if the Silhouette score of the larger topic would be improved by merging with the smaller topic.

5.1.6 Filtering Topics after Percolation

After graph percolation, it is possible for small kernels to persist, having merged with no neighboring kernels. In order to reduce noise, and promote a more coherent topic set, kernels of size two (2-cliques) are removed from the topic set (line 16).

5.1.7 Optional Post-processing: Augmenting Topics with Domain Language Model

In order to build larger topics that maintain high coherence, we train a Word2Vec embedding model [63]. We then use our locally trained embeddings to add words to

each topic after the topics have been found in the graph. We do so by computing the top n closest words to each topic word in the embedding space, and adding those n words to topic z_i, only if their addition maintains a graph property value of at least λ. We refer to this extended algorithm as λ-CLIQ$^+$.

5.2 PTM Empirical Evaluation

In this section, we present experiments performed on three different pre-labeled, domain-specific Twitter data sets to demonstrate the accuracy of our model.

5.2.1 Experiment Setup

Data Sets. Our first data set, which we refer to as the Political data set, contains 764,993 tweets about Donald Trump during the 2016 Presidential Election and Transition Period prior to his inauguration, recorded between August 2016 and December 2016. 5,000 tweets each day were selected at random during this period from all of the tweets that mentioned the candidate. Our second data set, which we refer to as the Parenting data set, contains 953,115 tweets about parenting collected between March 2016 and September 2017. The tweets were collected from the accounts of Twitter users identified as 'parenting authorities' by social scientists studying the affect of social media on parents [83]. The third and final data set, which we refer to as the Covid, contains 883,990 English tweets from between January 16th, 2020 and April 1st, 2020. Tweets were collected if they mentioned hashtags associated with the coronavirus pandemic, including '#covid,' '#coronavirus,' and '#covid19.'

Baseline Algorithms. We test our model against baseline models that represent models from each class of topic models, as discussed in section 2. The models that we

test against are Latent Dirichlet Allocation (LDA) [12], Hierarchical Dirichlet Process (HDP) [91], Non-Negative Matrix Factorization (NMF) [47], Topic Segmentation (TS) [30], Dirichlet Multinomial Model (DMM) [71], Self-Aggregated Topic Model (SATM) [79], GPU-DMM [54], and Biterm Topic Model (BTM) [103]. For LDA, TS, and BTM, we used k=20, 100, and 30 topics, respectively. All other baseline models produced their best topic sets at k=50. These values were chosen empircally after testing each model on k-values ranging from 10 to 100. All other parameters were left at their suggested values.[1] We leave a more comprehensive sensitivity analysis for future work.

Data Preprocessing. The manner in which data is preprocessed can impact topic modeling results. Given the noisy nature of Twitter data, we consider this to be as important to the construction of our model as any of the other steps. We employ the preprocessing pipeline as defined by Churchill et al. [23], and use the following preprocessing steps, in the following order: Remove deleted posts, remove user tags, remove urls, remove punctuation (including hashtag symbol), lowercase text, remove stopwords, stem words, remove words of length less than three.

Deleted posts, user tags, and urls all contribute to noise while adding no rich topic content. We remove these prior to removing punctuation because they are easily identified with punctuation still intact. We remove stopwords, again to reduce noise, and we stem words to reduce the size of the vocabulary and get higher average word frequency. We compared lemmatization and stemming and found their performance in terms of topic quality to be similar. We chose to use stemming in our preprocessing. Finally, we remove short words in order to reduce noise.

The data set statistics are shown in Table 8.1. $|D|$ represents the number of tweets. The final two columns show the difference in vocabulary size before and after prepro-

[1]We use the Mallet implementation of LDA

Table 5.1: Data Set Vocabulary Size before and after Preprocessing

Data Set	|D|	All Tokens	Unique Tokens in D	Unique Tokens in D''
Political	764,993	14,404,344	931,572	380,793
Parenting	953,115	14,658,377	1,562,491	522,797
Covid	883,990	13,766,991	855,734	791,014

cessing. The difference in the starting and ending vocabulary for Covid is signficantly smaller than those of the Political and Parenting data sets because the unprocessed Covid data set did not contain URLs.

Constructing the Phrase Model. Once our data set has been cleaned, we approximate phrases by creating ngrams up to 4-grams from the remaining clean text. Other more traditional means of identifying phrases, such as Named Entity Recognition and Noun Phrase detection, are less attractive options for social media data because they rely on accurate part-of-speech tagging and well-formed sentences to produce accurate results. We then replace the component words of ngram with the respective ngrams within each document. We identify frequent ngrams using the Natural Language Tool Kit (NLTK) [9]. We empirically evaluated our parameters and determined that setting the minimum frequency of an ngram to be 256 is reasonable for all of our data sets. The higher the minimum frequency, the faster the construction of the Phrase model, but the smaller the vocabulary. Too small a vocabulary is problematic, so balancing graph construction cost and vocabulary size is important.

Training the Word Embeddings. In order to find closely related words for improving topics, we train a Word2Vec model using CBOW on our data set [63]. A skip-gram model would be an acceptable substitute for CBOW here. We train our model with 100 features, using the Gensim Python library [80].

Algorithm Parameters. We evaluate two variants of the λ-CLIQ: using density (λ-CLIQ$_D$) and using Silhouette (λ-CLIQ$_S$) during percolation. We also evaluate λ-

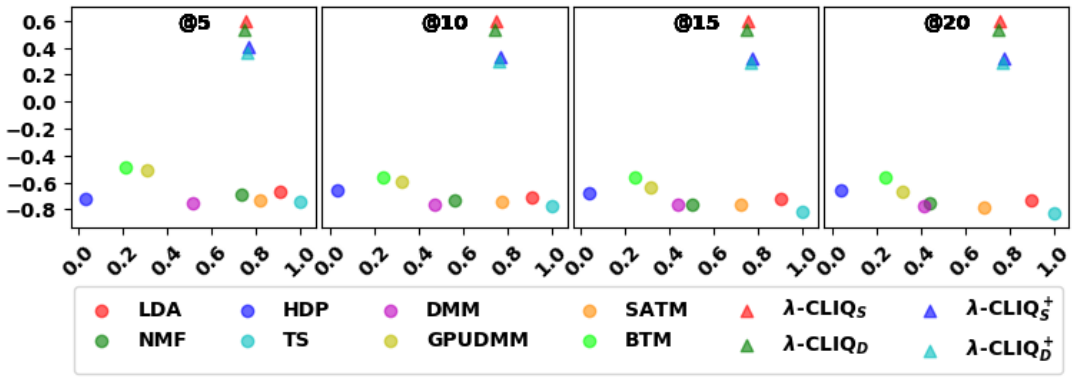


Figure 5.2: Coherence Compared with Diversity for each Model on the Covid Data Set.

CLIQ$^+$. While we conducted an extensive sensitivity analysis, we present only the parameters for the best results in the paper. For the Political data set, $\tau = 1$, and the embedding distance threshold = 0.5. For the Parenting data set, $\tau = 50$, and the embedding distance threshold = 0.25. For the Covid data set, $\tau = 1$, and the embedding distance threshold = 0.25. These parameters remain the same for all three algorithm variants.

5.2.2 Quantitative Analysis

In this section, we perform a quantitative analysis of our model's performance against the baseline models. We consider two metrics, topic coherence and topic diversity, that together attempt to address how well models can isolate meaningful topics in the data sets.

Comparison. As discussed in Section 3.2, topic model with good coherence and diversity scores is likely a better model than one with only a high score in one of the two categories. We present this relationship in Figures 5.2, 5.3, and 5.4.

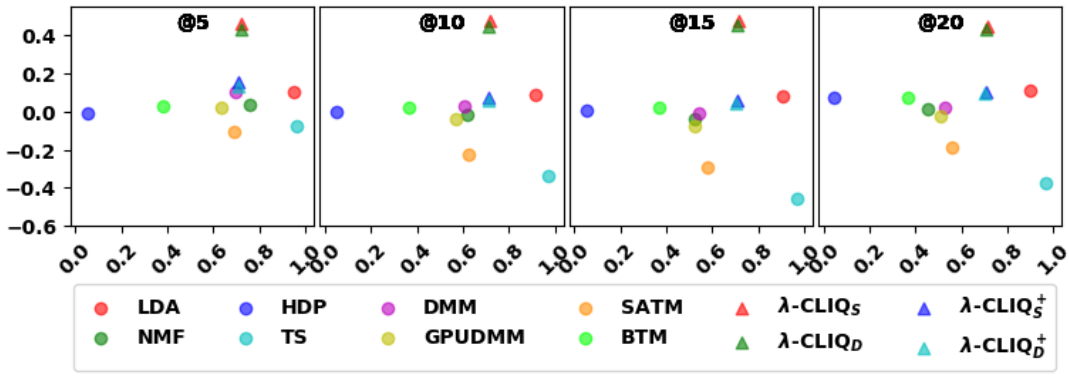


Figure 5.3: Coherence Compared with Diversity for each Model on the Parenting Data Set.

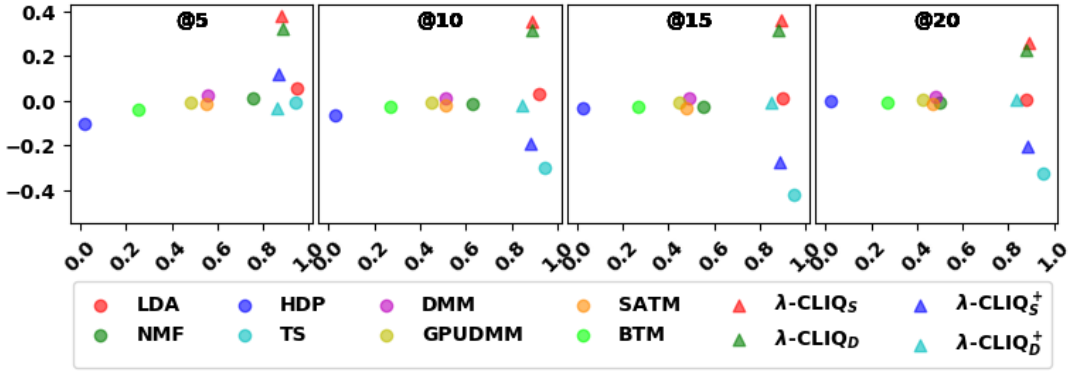


Figure 5.4: Coherence Compared with Diversity for each Model on the Political Data Set.

Each plot compares the coherence score (y-axis) with the diversity score (x-axis) of each model for the top-x words in each topic. The number of top words is indicated at the top of each plot (@5, 10, 15, 20). The baseline models are represented by circles, whereas the variations of our model are represented by triangles. By visualizing the results in this way, we can easily tell which models produce the best topics according to coherence and diversity. The best methods having high coherence and high diversity would be in the top-right corner of each plot.

The first observation is that λ-CLIQ$_S$ and λ-CLIQ$_D$ are closest to the upper right corner for all data sets across all top word sizes. In all data sets, we see that λ-CLIQ$_S$

produces slightly better coherence than λ-CLIQ$_D$. The extended variations have a lower coherence than their counterparts without embedding augmentation. This is due to the nature of embedding augmentation. Adding more words to a topic can result in lower NPMI due to the added words not having high co-frequencies with all of the original topic words. This is best visualized in the Political data set results, where the NPMI of λ-CLIQ$^+{}_S$ drops drastically from the @5 to @10 plot. This indicates that the set of words added by embedding augmentation contained significant amounts of noise. Recall that the other two data sets were set to a lower embedding augmentation threshold, and their NPMI scores do not drop as significantly. For the extended variants, λ-CLIQ$^+{}_S$ still edges out λ-CLIQ$^+{}_D$, with the exception of the Political data set.

The best baseline model overall is LDA, which has coherence competitive with the other models, and a much higher diversity than other baselines with high coherence. In the Covid data set, the baseline models all have very poor coherence, while our models get the highest coherence scores for any data set. It's unclear as to why the baseline models have such poor performance, but it is possible that the poor performance stems from the high frequency of generic words (flood words) about coronavirus. If most people talk about how coronavirus affects their everyday life, then it is possible that more generic words, that share little mutual information with all other words, are found to be the top words by the baseline models. The proliferation of generic words in the topics would explain the low NPMI scores. Another reason that Covid could be a more difficult data set than the other two is its larger vocabulary size, meaning that cofrequencies will on average be lower.

In the Parenting data set, we see a reasonable comparison between our models and LDA. The extended variants produce topics with a similar coherence, but lower

λ-CLIQ $_S$	λ-CLIQ^+_S	LDA	HDP	NMF	TS	DMM	GPUDMM	BTM	SATM
julian$assang	**julian$assang**	elect	donald	show	**hillari**	**hillari**	donald	donald	campaign
wikileak	**wikileak**	**russia**	**hillari**	realiti	media	**wikileak**	**email**	peopl	report
clinton$campaign	**clinton$campaign**	**putin**	support	lose	still	fake	**hillari**	trump	return
trump$campaign	trump$campaign	**email**	vote	**email**	actual	**leak**	accus	**clinton**	call
manag	manag	report	**clinton**	ahead	candid	**email**	**wikileak**	campaign	press
	snowden	**russian**	peopl	**leak**	press	**clinton**	**clinton**	think	**hillari$clinton**
	hillaryemail	releas	elect	world	blame	break	break	media	attack
	modi	return	obama	real	corrupt	donald	media	american	trump
	podestaemail	white$hous	presid	return	**email**	democrat	video	**email**	time
	podesta	prove	media	record	sick	video	**leak**	show	scandal
	dncleak	**wikileak**	trump	late	deplor	report	fake		hide
	clinton$camp	**leak**	campaign	crowd	creat	media	report		health
	robbi	offici	said	night	**wikileak**	plan	call		**leak**
	mook	confirm	call	video	tcot	campaign	stori		watch
	lewandowski	evid	think	respect	liar	elect	miss		
	donald$campaign	hide	make	daili	**hillaryclinton**	accus	**russia**		
	donald$trump$campaign	told	time	today	slam	expos	attack		
	payrol	**hack**	america	tonight	push	stori	show		
		msnbc	need	1	**clinton$camp**	2016	tape		
		deni	women	**wikileak**	wake	support	claim		

Figure 5.5: A Topic about the Leak of Hillary Clinton's Emails via Wikileaks, Captured by each Topic Model. Bolded Words are Highly Related to the Topic.

diversity, while the non-augmented variations get a much higher coherence and similar diversity to that of their augmented counterparts.

With the non-augmented variants of our model producing better coherence scores than our extended variants, a natural question is why bother with the embedding augmentation at all? We address this with our qualitative analysis.

5.2.3 Qualitative Analysis

Similar to clustering, determining which topics are the most interpretable does not always map to the ones with the highest quantitative evaluation. Therefore, we show a qualitative case study comparing a very popular topic from the Political data set across all models. We chose a topic from the Political data set specifically because λ-CLIQ^+_S gets a poor coherence score in that data set. The topic that we chose pertains to the scandal surrounding the leak of Hillary Clinton's private emails, orchestrated by Russia and published by Wikileaks.

Figure 5.5 shows this topic as it was captured by each topic model in our experiments. For the sake of space, we limit each topic to the top twenty words, although

some are shorter. In this figure, the words most relevant to the topic (as decided by comparing to expert-labeled topics) are bolded. We can see from a glance that the topic model with the most relevant words is λ-CLIQ$^+{}_S$. It contains ten bolded words out of eighteen, indicating that it successfully isolated a significant number of relevant topic words. From a qualitative perspective, having email and wikileaks is important for this topic. Four out of eight of the baseline models have those two words. LDA, TS, DMM, and GPUDMM all produce reasonable topics, but each model allows much more noise than λ-CLIQ$^+{}_S$.

A disadvantage of the generative baseline models (LDA, HDP, DMM, GPUDMM, BTM, SATM) compared to the percolation-based models is that their statistical nature dissuades the inclusion of ngrams in the most likely words of a topic. Because the models hinge on high frequency and high co-occurrence between tokens, unigrams are much more likely to have a higher weight than ngrams, which are much less frequent. In the percolation-based models, we see far more ngrams in each topic, many of which are relevant topic words.

Comparing Coherence Scores. Returning to the question of why we bother using embedding augmentation, let's consider the topic generated by λ-CLIQ$^+{}_S$. In this example, it is clear exactly which words came from the embedding augmentation, and which came from the standard model. We can see that while the first three words in the topic from λ-CLIQ$_S$ capture important facets of the topic, the words from the embedding augmentation provide important context such as *hillaryemail*, *podestaemail*, *dncleak*, and *lewandowski*. The 'words' containing *email* are hashtags that were deconstructed in the preprocessing phase, and give the context of what was released on Wikileaks. Along with *lewandowski*, they also provide the subjects of the scandal, presidential candidate Clinton and her campaign executives. *dncleak*

provides further context about relevant events associated with the scandal during the campaign period.

When evaluating topic models, quantitative evaluation alone is not sufficient. Conducting both a qualitative and a quantitative analysis gives us more insight into the quality of topics generated by different models. By doing so, we are able to see the value of adding embedding augmentation to our model.

Chapter 6

Topic-Noise Models: TND and NLDA

In creating the Percolation-based Topic Model, we found that it was indeed possible to remove most context-specific noise from topics. The problem, however, was that PTM was inconsistent in the number and size of topics that it created. Often, topics consisted of only a handful of words, and augmenting topics with word embeddings reintroduced a significant amount of both types of noise.

We turned our attention back to generative models, which we knew to have problems with noise inundation in social media data. The advantage of generative models is that they produce a predictable number of topics, and every word in the vocabulary has a probability of being in each topic. The top-n most likely words per topic make up what we see as topics. Our goal now was to produce highly interpretable topics by keeping noise words out of the top-n most likely words per topic. As we showed in Chapter 4, preprocessing can help with context-independent noise, but we needed a way to remove context-specific noise.

There are a few state-of-the-art models that attempt to improve the coherence of topics in short texts such as social media. The first two are DMM [109] and GPUDMM [54], which are mentioned in Chapter 4. The third is Common Semantics Topic Model (CSTM) [56]. CSTM adds 'common topics' to the mixture of topics from which documents are generated. Similar to DMM, a document can be generated from a single traditional topic. However, it can also be generated from any of the common topics, which try to capture noise words.

At the beginning of our attempts to remove noise from generative models, we made an important observation about noise in topic models: Context-specific noise is dependent on the topic set that is generated. For example, the word 'Hogwarts' certainly belongs in a topic set generated from the Harry Potter books. However, in a topic set generated from Covid-19 tweets, it would probably be noise.

This observation led us to the realization that we must jointly generate a topic and noise distribution in order to get an accurate representation of context-specific noise words. To detect noise, it is not enough to observe word frequencies and other qualities prior to topic generation, nor is there a foolproof way of postprocessing that would remove the correct words. We defined a new type of model, the *topic-noise model*, to be a model that jointly approximates the noise and topic distributions of a data set. Our topic-noise model, Topic-Noise Discriminator (TND) [22], is detailed and analyzed in the rest of this chapter.

6.1 Designing Topic-Noise Models: Topic-Noise Discriminator (TND) and Noiseless LDA (NLDA)

In order to relate our models to the most relevant in the previous literature, we begin by presenting the plate notation and describing LDA [12] and (SWB) [20] (Section 6.1.1). We then describe our proposed topic-noise model (TND) (Section 6.1.2), and the extension using embedding sampling (Section 6.1.3). Finally, we describe our approach for combining existing generative and non-generative models with the noise distribution generated by TND (Section 6.1.4).

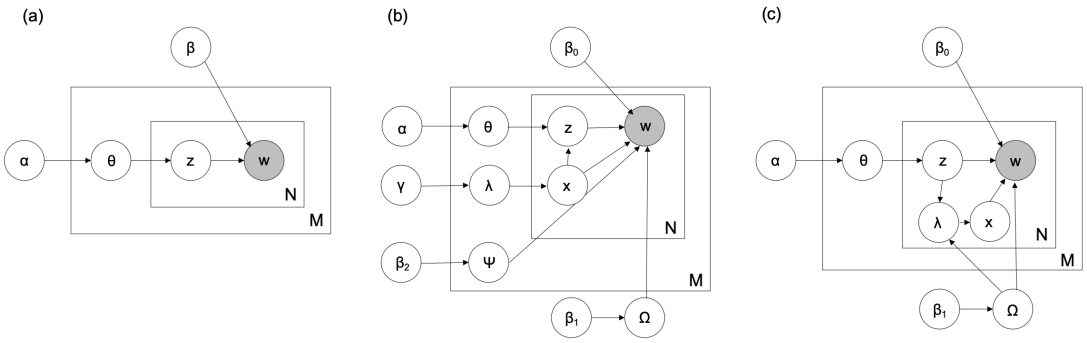


Figure 6.1: LDA (a), SWB (b), and TND (c) Generative Models.

6.1.1 LDA and SWB Topic Models

Figure 6.1 shows the graphical representations of LDA (a) and SWB (b). While the entire generative process for LDA is presented by Blei et. al [12], we present the high-level generative process in our notation here.

For $d \in D$:

1. Draw the number of words N for d.
2. Draw the topic distribution θ from the Dirichlet distribution, conditioned on the parameter α.
3. For each word w_i, $0 \leq i < N$:
   (a) Draw a topic z_i from θ.
   (b) Draw a word w_i based on the probability of w_i given the topic z_i and conditioned on the parameter β.

The special words topic model with a background distribution (SWB), proposed by Chemudugunta et al. [20], improves on LDA by adding a special words distribution for each document, and a global background distribution. SWB's generative process works similarly to that of LDA, but with some important changes to account for its

extra distributions. First, a word is not guaranteed to be drawn directly from the document's topic distribution. Instead, it can be drawn from the document's topic distribution, from the document's special words distribution (Ψ in Figure 6.1(b)), or from the independently computed global background distribution (Ω in Figure 6.1(b)).[1] The decision of which distribution to draw from is controlled by x, which is sampled from a document-specific multinomial λ conditioned on γ.

6.1.2 Topic-Noise Discriminator (TND)

Recall that we define a document as a mixture of topics and noise. Therefore, our generative model, Topic-Noise Discriminator (TND) alters the generative process of the topic distribution to account for an underlying noise distribution. The graphical model for TND is shown in Figure 6.1(c). We identify noise by approximating the distribution of noise words across the document collection D.

Intuitively, instead of each word in the document being drawn from the document's topic distribution (as in LDA), each word is drawn from *either* that document's topic distribution, *or* a global noise distribution, based on the probability of the individual word being in a topic or in the set of noise words. While this looks similar to the special word distribution in SWB, it is designed differently. SWB is designed to capture words that appear in a specific document and rarely anywhere else. The underlying assumption here is that these special words appear frequently in their respective documents, such as the word 'Hogwarts' would appear an irregularly high number of times in a Harry Potter book, and almost never in other contexts.

In social media data, documents are so small that with high certainty, words will not appear frequently enough in a single document for them to affect the composition

[1]The global background distribution is not the same as our noise distribution, but the notation is the same.

of an entire topic, and any word that appears in a single document will be removed by reasonable preprocessing (such as removing words that appear only once in the data set). Therefore, the special words distributions are not needed for a topic model that is intended for social media data because that distribution cannot capture the 'right' words, thereby unnecessarily complicating a model designed for short posts. The background distribution is closer to how we model the noise distribution. However, the SWB background distribution is computed independently. In contrast, our noise distribution is not.

The decision of whether a word is a topic word or a noise word is determined using the Beta distribution (see Figure 6.1). The Beta distribution, λ, is the special case of the Dirichlet where k=2, and x is the switching variable controlling whether the word is drawn from z or Ω. This distribution is conditioned on the γ parameter. Setting the initial value of γ higher allows us to skew the distribution to favor topics if the expectation of noise is less than topics. In practice, using the Beta distribution helps produces topics that contain far less noise than traditional generative models such as LDA. Equation 6.1 shows the calculation of the Beta distribution for each word. The Beta distribution takes into account the topic frequency and noise frequency of the given word. Using the square root of the word's frequency in the topic and noise distributions reduces the likelihood of a word continually moving between topics and noise. The effect of this alteration in the generative process is that over many iterations, noise words slowly start to affect document-topic assignment less and less.

$$Beta(\sqrt{\beta_z^i + \gamma}, \sqrt{\Omega_i}) \tag{6.1}$$

The noise distribution is not a static list, like stopwords, nor is it a strictly frequency-related list like TF-IDF rankings. Instead, the noise distribution is generated with respect to a set of topics simultaneously being generated on the data

set. As such, the noise distribution has knowledge of topic words baked into it, as opposed to approaches that attempt to identify noise words without approximating a topic-word distribution.

The generative process for TND is as follows.

For $d \in D$:

1. Draw the number of words N for d.
2. Draw the topic distribution θ from the Dirichlet distribution, conditioned on α.
3. For each word w_i, $0 \leq i < N$:
   (a) Draw a topic z_i from the topic distribution θ.
   (b) Draw a word from either z_i or the noise distribution Ω, according to the Beta distribution, conditioned on α.
   (c) If drawing from z_i, draw w_i based on the probability of w_i given the topic z_i and conditioned on β
   (d) If drawing from Ω, draw w_i according to the probability of w_i given Ω and conditioned on γ.

6.1.3 Embedding Sampling

With recent advances in natural language processing, we propose using word embedding vectors to increase the probability of semantically related words appearing together in specific topics and in the noise distribution. GPUDMM, proposed by Li et al. [54], uses word embeddings in a similar fashion, altering the traditional Gibbs sampling algorithm so that whenever a word is sampled, words related to it in the given embedding space are also sampled. In Gibbs sampling, one word is sampled at a time. In Generalized Polya Urn (GPU) embedding sampling, the word is returned

with other similar words. This increases the likelihood of related words being in the same topic.

This is a clever way of producing more coherent topics, but in social media, this also allows for noise words to pull even more noise words into topics. However, using this same sampling scheme within TND, where noise words are modeled in their own distribution, we should see noise words pulling more noise words into the noise distribution instead.

To ensure that we do not pull the wrong words into the wrong distributions, we wait τ iterations to begin GPU embedding sampling. After τ iterations, and every τ iterations thereafter, we re-evaluate the words eligible for GPU embedding sampling. Only words whose probability of being in the noise distribution or of being in a single topic is higher than ν standard deviations from the average are considered. By narrowing words down this way, we ensure that we do not pull the related words of low-probability words into topics.

Our sampling approach allows for the scaling of the impact of embeddings on TND. By setting the parameter $\mu \geq 0$, we can decide how many related words to sample for each word in GPU embedding sampling. Setting $\mu = 0$ is equivalent to traditional Gibbs sampling, while increasing μ means more and more impact of embeddings on the model.

6.1.4 Extending Existing Topic Models with TND

The noise distribution generated by TND can be integrated into any topic model that produces a topic-word distribution, as generative models do. By comparing a word's probability in a topic and in noise, noise can be efficiently filtered from a topic set, leaving more coherent, interpretable topics with little overhead. We show this approach here, combining TND and LDA to create NLDA.

Noiseless LDA (NLDA). While TND produces topics, it also provides a useful noise distribution that can be easily transferred to other topic models. In the case where we have a pre-trained topic model that uses a topic-word distribution to approximate topics, we can apply the pre-trained noise distribution from TND in an ensemble to probabilistically remove noise words in a similar manner to the process within TND. In Noiseless LDA (NLDA), we borrow the noise distribution generated by TND, and use it with LDA, thereby creating a version of LDA that contains topics with fewer noise words.

To create NLDA, we train a noise distribution Ω on D using TND, and we train an LDA model on D.[2] We then produce a topic set by combining the noise distribution of TND and the topic-word distribution of LDA. Similar to deciding whether a word is a topic or noise word, for each topic $z \in Z$, we remove w_i from z according to a Beta distribution (Equation 6.2) conditioned on w_i's frequency in noise and in LDA's topic distribution.

In order to make noise distributions more transferable to different parameters of LDA, we add a topic weight parameter ϕ to the Beta distribution calculation to downsample or oversample the noise distribution. Equation 6.2 shows how ϕ is used to scale the noise distribution based on k, the number of topics in the LDA model.

$$Beta\left(\sqrt{\beta_z^i + \gamma}, \sqrt{\Omega_i(\phi/k)}\right) \tag{6.2}$$

For each word w_i in topic z, once we have determined its status using the Beta distribution, we take one final step to facilitate better topic filtering. If w_i is removed from z, w_i's frequency in the noise distribution is incremented, marking it as noise once again. If w_i is retained in z, w_i's frequency in the noise distribution is increased by β_z^i. By increasing w_i's noise frequency *after* it is included in a topic and maintaining

[2]The k value does not have to be the same for the two models.

the topic frequency, we are deterring its inclusion in future topics, which share the noise distribution. In this way, through the Beta distribution (Equation 6.2), we have increased the relative probability of future topics determining it to be noise.

Decreasing ϕ to a value lower than k ($\phi < k$) will result in a lower beta value, and therefore less harsh noise filtering, while increasing ϕ to a value greater than k ($\phi > k$) will result in a higher beta value, and harsher noise filtering. Setting $\phi = k$ results in an unweighted NLDA. The addition of ϕ allows for NLDA to be scaled to larger data sets and different values of k using the same original noise distribution. While this will be unnecessary for many use cases, the ability to essentially transfer a noise distribution to different parameter settings makes NLDA more usable and faster. It also requires less storage during model construction.

Context Noise List Usage. Not all topic models produce topic-word distributions, and often we have access to only a set of topics that we would like to filter noise from. In the case where we have a pre-trained topic model that does not use a topic-word distribution to approximate topics, or in the case where we have only a set of topics, we can apply the TND noise distribution in a more crude manner, using a context-specific noise list. In this approach, which we call Context Noise List Usage, we propose filtering words from a topic set that have a high probability in the noise distribution. For a given noise distribution Ω, we define Ω_c to be the set of c words in the noise distribution with the highest probabilities. For each topic $z \in Z$, we remove word w_i from z if $w_i \in \Omega_c$.

This approach is more likely to remove flood words than lower-frequency noise words, but it can still be beneficial to topic sets. We will demonstrate this in the next section.

6.2 TND and NLDA Empirical Evaluation

In this section, we present our empirical evaluation. We evaluate the three variants proposed in Section 6.1: Topic Noise Discriminator (TND), Noiseless LDA (NLDA), and Context Noise List Usage for existing models. We begin by describing our experimental setup, including a description of the data sets, the preprocessing, and the model parameters (Section 6.2.1). We then present our quantitative evaluation (Section 6.2.2), followed by our qualitative analysis (Section 6.2.3).

6.2.1 Experiment Setup

Baseline Algorithms. We compare our proposed models to the following state-of-the-art models: Latent Dirichlet Allocation (LDA),[3] Gibbs Sampling Dirichlet Multinomial Mixture (DMM) [109], Generalized Polya Urn Dirichlet Multinomial Mixture (GPUDMM) [54], and Common Semantics Topic Model (CSTM) [56]. These topic models each represent a unique facet of generative topic models as explained in Section 2. As mentioned in the previous section, because SWB is designed with fewer longer documents in mind, the computation cost is too high for large volumes of social media posts and the special words distribution is not meaningful for the short post environment. We also compare our graph-based model, PTM, to our models on the two larger data sets that we test on.

Data Sets. In this analysis, we consider four data sets: a newsgroup data set, two Covid-19 data sets, and an election 2020 data set. Our first data set is a subset of the Twenty Newsgroups data set [52]. We use the training set, containing 11,024 documents, to assess how well the different models generate topics that map to the labeled data. While 20 Newsgroups is a relatively small data set, it provides a platform

[3]Specifically the MALLET implementation of LDA [60]

for reproducibility and allows us to see the impact of our algorithm on a data set that contains less noise than traditional social media data sets.

We also have two Twitter data sets. The first data set contains posts about the 2020 Covid-19 pandemic. Using Covid-19 related hashtags, we collected Covid-19 related tweets through the Twitter Streaming API. For this analysis, we consider two samples of these data. The *50k Covid-19* data is a random sample of 50,000 tweets about the 2020 Covid-19 pandemic, collected between mid-January and April 2020, a time period of massive change in the conversations revolving around the pandemic. The *Million Covid-19* data contains over 1 million tweets about the 2020 Covid-19 pandemic, collected between August 1 and September 30.

The other Twitter data set, *Election 2020*, contains posts about the 2020 United States Presidential election. Using relevant hashtags and keywords, we collected these data between January 1 and September 30 through the Twitter Streaming API. This data set consists of over 1.4 million tweets, focusing on topics related to the November election. Both the Million Covid-19 and Election 2020 data sets can be used to test the ability of the different models to produce high-quality topics on larger, noisier social media data sets.

Data Preprocessing. Data preprocessing can have a significant impact on topic models [23]. For each of our Twitter data sets, we remove deleted posts and remove user tags. For all of our data sets, we lowercase text and remove urls, punctuation (including hashtags), and stopwords.

Model Parameters. In order to provide a thorough sensitivity analysis for each of our models, we test each model with many different parameter settings.[4] Because of space limitations, we only present the results for the best performing models. For

[4]Parameters for sensitivity analysis across models: $k = 10, 20, 30, 50, 100$; $\alpha, \beta_0 = 0.01, 0.1, 1.0$; $\beta_1 = 0, 16, 25, 36, 49$; $\phi = 5, 10, 15, 20, 25, 30$; $\mu = 0, 3, 5, 10$

TND, the best parameters for producing its own topic set were $\alpha = 0.1$, $\beta_0 = 0.01$, $\beta_1 = 25$, $k = 30$, $\mu = 0$, and $\nu = 1.5$. However, the best noise distributions for use in NLDA occurred when $\mu > 0$. For NLDA, the best performing parameters are $\alpha = 0.1$, $\beta_0 = 0.01$, $\beta_1 = 25$, and $k = 30$. As we will see, the best parameter for μ and ϕ varied based on the data set. We found that β_0, α, and β_1 were far more stable parameters and that changes in their values did not have significant effects on the performance across data sets. μ and ϕ cause more noticeable effects on performance based on the data set. In the case of ϕ, tuning is quick in practice because it applies to the ensembling of TND and LDA, where values of ϕ can be quickly iterated through on the trained models. For LDA, they were $\alpha = 0.1$ and $\beta = 0.01$. For DMM and GPUDMM, we found $\alpha = 0.1$, $\beta = 0.1$ to be the best parameters. For GPUDMM, we used GloVe Twitter word embeddings [74] with both 50 and 100 dimensions, and found the difference in topic quality to be negligible. The results shown here use 50 dimensions. For CSTM, we used the suggested settings for nu_f and nu_c, 1 and 0.1, respectively. We found $\alpha = 0.1$ and $\beta = 0.01$ with 2 common topics to be the best parameters. While the other settings tested did reduce the quality of the topics obtained, their results were similar. For PTM, we used the λ-CLIQ$^+{}_S$ version, with topic augmentation using word embeddings. For the Election 2020 data set, we used $\tau = 1$, and the embedding distance threshold = 0.5. For the Covid-19 data set, we used $\tau = 1$, and the embedding distance threshold = 0.25. These were the best settings for the similar-type data sets tested on in Chapter 5.

6.2.2 Quantitative Analysis

In this section, we use topic coherence and topic diversity to compare the different topics generated from either topic models or topic-noise models.

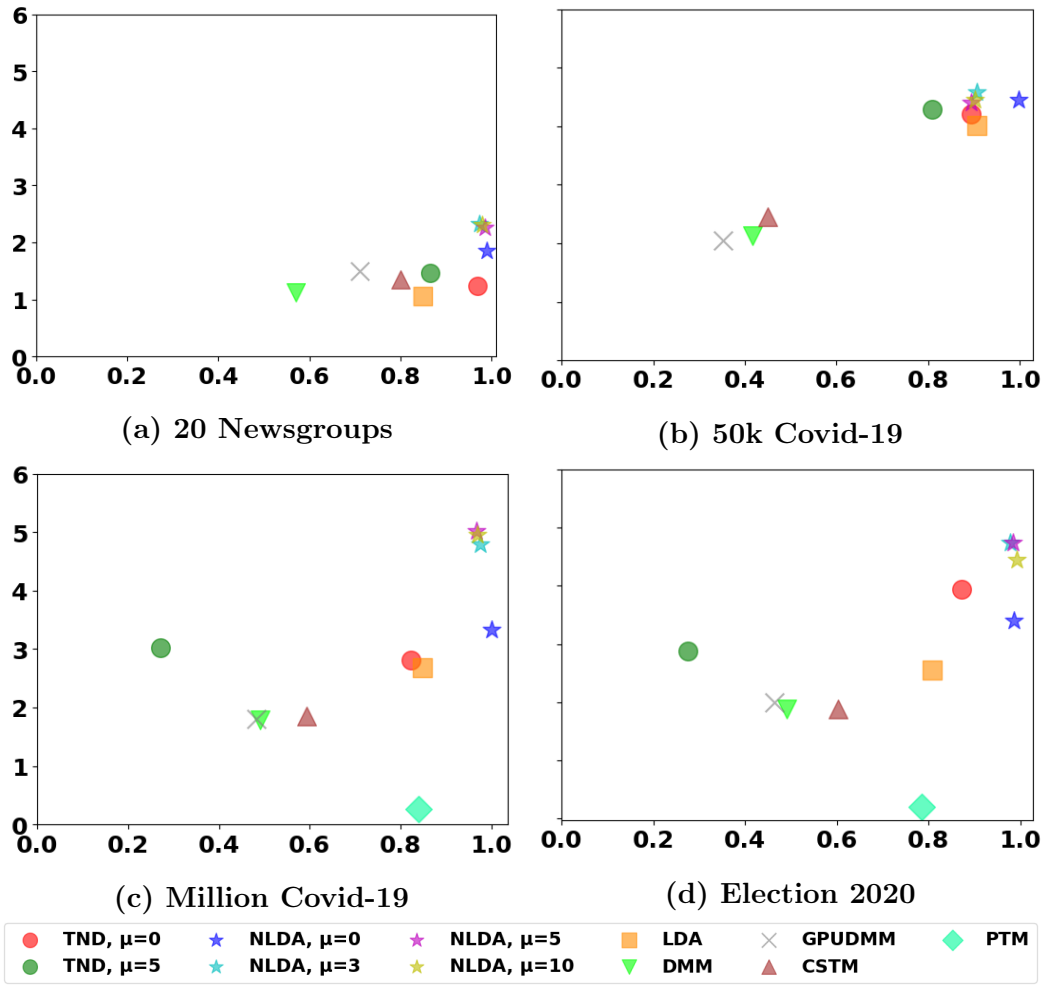


Figure 6.2: Comparison of TND and NLDA to Baselines. Coherence (y) and Diversity (x). $k = 30$. $\beta_1 = 25$ for TND

We begin by comparing the performance of models on the 20 Newsgroups data set. Figure 6.2a shows the coherence and diversity of each model. On the x-axis is topic diversity, and on the y-axis is topic coherence. The models closest to the top right corner of the plot have the best topic coherence and topic diversity. Figure 6.2a shows that NLDA is clearly the best model for both topic coherence and topic diversity. GPUDMM and TND ($\mu = 10$) have the second best topic coherences, and TND ($\mu = 0$) has the second best topic diversity. Of all the data sets, this one contains the least amount of noise. It is interesting that in this context, using the estimated

noise distribution from TND within NLDA leads to stronger results than LDA alone or estimating both the topic and noise distributions together in TND. This highlights that even in a less noisy data set, modeling noise is important. We surmise that GPUDMM performs well on this data set because the number of words is smaller and the context of words is more stable in newsgroup data.[5]

Next, we compare the results of the best settings for each model on the 50k Covid-19 data set (Figure 6.2b). Again, topic diversity is plotted on the x-axis, while topic coherence is on the y-axis. On the left, we can see a cluster of the DMM, GPUDMM, and CSTM results. All three models produce topic sets with similarly low topic coherence and topic diversity. TND produces more coherent and diverse topics than DMM, GPUDMM, and CSTM. LDA produces similar results to TND. However, NLDA is the best model overall. In other words, first building the topics using the context-specific noise words and then using the estimated noise distribution to iteratively reduce the noise in LDA topics improves the topic coherence by 5.6% over TND and 10.5% over LDA. It also increases the topic diversity by 11.6% over TND and 10% over LDA.

We pause to reflect on the fact that the topic coherence scores for the Twitter data sets are much higher than the newsgroup data set. We think this is a result of the sparsity of the Twitter data. The percentage of high frequency words in the newsgroups that are not flood words is much higher than in the Twitter data, leading to more words overlapping across topics than for the Twitter data. This highlights the importance of separating high frequency content-rich words from high frequency content-poor words.

[5]A natural question here would be, given that there are 20 newsgroups, why not use $k = 20$? We found that every model produced better results with $k = 30$.

In order to show that these models are effective on larger data sets, we show the results of our models on the Million Covid-19 and Election 2020 data sets, compared with the results of the best-performing baseline models. While TND is slower than LDA, it is still considerably faster than other models that attempt to account for noise distributions and embedding spaces, like CSTM. With this in mind, we use this section to show the transferability and reusability of TND's noise distributions and how NLDA's ϕ parameter allows us to easily adapt a noise distribution to any number of topics. The results we present use the following parameters for TND: $\alpha = 0.1$, $\beta_0 = 0.01$, $\beta_1 = 25$, $k = 30$, $\nu = 1.5$, and $\mu = \{0, 3, 5, 10\}$. We tested NLDA on $k = \{10, 20, 30, 50, 100\}$ and $\phi = \{5, 10, 15, 20\}$, but show only the best parameter settings for clarity.

Figure 6.2c presents the topic coherence and topic diversity of the models built using the Million Covid-19 data set. In Figure 6.2c, topic diversity is plotted on the x-axis, and topic coherence is on the y-axis. Again, NLDA produces results with consistently high topic coherence and topic diversity across k values with $\phi = 10$. It is clear here that for TND, using $\mu > 0$, meaning incorporating the embedding space to some extent, improves the coherence of NLDA substantially. However, as a standalone model, TND is far more coherent when $\mu = 0$. TND alone is always at least as good as LDA, and also produces a noise distribution that can be used by researchers to better understand the context-specific noise present in their data sets. NLDA's coherence improvement over its competitors is amplified on the Million Covid-19 data set. Its topic coherence increases by 19% over TND and 24% over LDA. It also increases the topic diversity by 21% over TND and 18% over LDA. The coherence of topics likely drops due to the size of the data set – as more documents are added to a data set, more words exist in the vocabulary, and the overall sparsity of the data set increases, thereby reducing the probability of words appearing together.

Figure 6.2d presents the topic diversity and coherence of the best models on the Election 2020 data set. NLDA again outperforms the field in both metrics, followed by TND. It is as good as NLDA in terms of coherence, and nearly as diverse. LDA is the next best model followed by CSTM. DMM and GPUDMM performed poorly for both topic coherence and topic diversity. This results because of the prevalence of context-specific noise in all of their topics. CSTM, another model designed to filter noise from social media texts, does get improved topic diversity compared to DMM on both the Election 2020 and Million Covid data sets, but it fails to produce more coherent topics.

We pause again, this time to look at the blue diamond in Figures 6.2c and 6.2d, signifying PTM. As we can see, its topic coherence is very low, despite gaining a diversity score of about 80% on each data set. This is in line with the coherence and diversity scores it received on the data sets in its initial testing in Chapter 5. The unusually low score is due not to its inability to find topics, but moreso to the propensity for topic augmentation using word embeddings to add more noise to topics. The two models most directly reliant on word embeddings, GPUDMM and PTM, both suffer from this problem. When it comes to a direct comparison of TND, NLDA, and PTM, a fair comparison is hard to make. PTM's nature is to find fewer, small topics that it is certain of, potentially adding noise in the final augmentation step, while TND and NLDA are generative models that find a chosen number of topics and actively remove noise in the final step. It seems that the latter approach is more successful in producing topic sets that resonate with our evaluation methods as well as with researchers.

Finally, we consider the noise penetration rate. We worked with social scientists and CNN researchers to develop a set of flood words (context-specific noise words) that were seen in open-ended survey responses about the 2020 presidential election.

Throughout the election cycle, as noisy words appeared in responses that detracted from semi-automated topic generation, they were added to the list. We use that expert curated list of 50 context-specific noise words to help understand noise penetration. While this does not represent a full set of noise words in the Twitter data set, these noise words are the bellwethers of noise that detracts from the specificity and meaningfulness of topics identified from short text responses like social media posts. Examples of context-specific flood words included `Trump`, `Biden`, and `people`.

Table 6.1 shows the noise penetration rate for the Election 2020 data set. TND contains almost zero noise, highlighting its namesake – noise filtering. Both TND and NLDA have a significantly smaller noise penetration rate than LDA, DMM, and even CSTM, the other model designed to reduce noise. In other words, our approach for reducing noise is able to effectively remove large amounts of noise, with an improvement in penetration rate of more than 0.8 when compared to LDA for the Election 2020 data set. Table 6.1 highlights the tradeoff that we make when we move from TND to NLDA. TND has a smaller level of noise penetration in topics. NLDA has more diverse and coherent topics, but with a little more noise penetration.

Context Noise List. In addition to showing TND and NLDA's success on modeling noisy data sets, we also show the effectiveness of the context noise list on topic sets produced by other topic models. As we observed in the previous analysis, GPUDMM underperforms in comparison to NLDA on the Twitter data sets, while performing well on the 20 Newsgroups data set. This is a direct result of the large amount of context-specific noise in the Twitter data sets. In this experiment, we will

Table 6.1: Noise Penetration in Election 2020 Data Set.

Model	LDA	DMM	GPUDMM	CSTM	TND	NLDA
Noise Pen. Rate	0.87	0.92	0.35	0.25	0.02	0.25

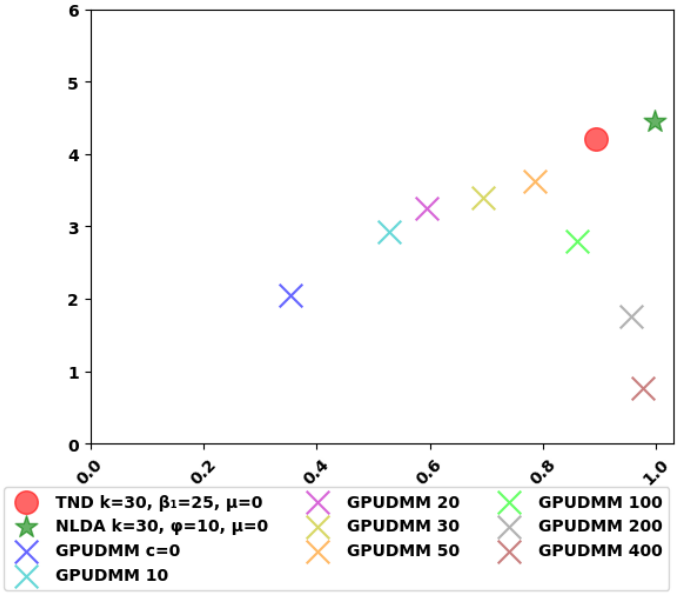


Figure 6.3: 50k Covid-19 GPUDMM with a Context-Noise List.

generate a context-noise list using TND and use it to filter words from generated topic lists.

Specifically, we fix $k = 30$ for TND, NLDA, and GPUDMM, and we use $\alpha = 0.1$, $\beta_0 = 0.01$, $\beta_1 = 25$, $k = 30$, and $\mu = 0$ as the parameters for TND to get an accurate noise distribution for use in NLDA and in the context-noise list. Figure 6.3 shows the impact of using a context-noise list of varying sizes with the GPUDMM topic set on topic coherence and topic diversity. Both TND ($\mu = 0$) and NLDA are shown for comparison purposes. We can see the topic diversity of GPUDMM increase as c increases, meaning that noise is to blame for much of the lack of diversity in the model. Even without incorporating the embedding space, the improvement in coherence is significant. When we look at topic coherence, we notice that when c gets very high ($c \geq 100$), the coherence of GPUDMM starts to fall off, even as its diversity continues to increase. In other words, removing small levels of context-specific noise can be useful for improving the topic coherence. Most of these words are flood words

that do not get removed through traditional avenues of preprocessing. For example, in the Covid data set, words that would be removed by the context-noise list include flood words like 'covid19,' 'coronavirus,' and 'covid,' and general noise words like 'people,' 'today,' and 'many.' Removing these words from topics will improve topic diversity and coherence by virtue of the replacements for these words being more informative for their respective topics. TND and NLDA are able to selectively remove only the noise words that are not closely tied to coherent topics, leading them to have higher topic diversity and topic coherence than models using the context noise list. However, we believe that researchers will still find it valuable to be able to remove context-specific noise when using models that are already part of their pipeline.

6.2.3 Qualitative Analysis

For the Election 2020 data set, human judges were asked to label topics from LDA, DMM, GPUDMM, CSTM, TND, and NLDA. Our evaluation was conducted by 18 people, 10 male and 8 female. Most judges were college students. Judges were presented with five 'selected topics' from the Election 2020 data set that were dominant topics during the campaign. Judges were asked to label topics generated by each of the models as one of the selected topics. If judges did not believe a selected topic was present, they could suggest another topic that applied, or they could indicate that no clear topic existed. Thirty topics from each topic model were used in the human judgment experiment. Based on our topic coherence and topic diversity results, we expected variation in terms of the number of topics that would be interpretable by human judgement. Each topic was labeled independently by three judges. In our results, we considered a topic successfully labeled only if all three judges agreed on its label since that provided the best results for the baselines.

Table 6.2: Fraction of Unique Topics Agreed on by Judges.

LDA	DMM	GPUDMM	CSTM	TND	NLDA
0.57	0.35	0.30	0.57	1.00	0.85

Table 6.3: Topic Labeling Judge Agreement.

Model	Vice President	Covid-19	QAnon	Debates	Mail-in Voting	Other
LDA	2	1	2	4	0	5
DMM	0	0	1	10	1	2
GPUDMM	1	0	1	9	0	2
CSTM	1	1	1	5	2	4
TND	0	1	0	0	0	3
NLDA	2	2	0	1	1	7

Table 6.3 shows the number of topic labels agreed upon by all three judges for each model. All the models except TND[6] had 13 or 14 topics that were interpretable and had topic agreement. This suggests that there is a possible upper limit on the number of topics a generative model can successfully detect for a given k parameter. Surprisingly, TND does not perform as well on the qualitative analysis in terms of topic agreement. In other words, even though it is one of the top models in terms of quantitative measures, that did not hold true for qualitative measures on our Election data set. However, removal of noise is clearly important since two of the top three models include noise removal. In terms of topic coverage, only CSTM had 100% (5/5) topic coverage of the specified topics, followed by NLDA and LDA with 80% (4/5). The other three models had poor topic coverage.

Next, we assess topic uniqueness. Some topics created by topic models are repetitive while others are more unique. Table 6.2 shows the fraction of unique topics returned. Here we see the real strength of both TND and NLDA. All the topics for TND are unique - none overlap. NLDA only labels two duplicate topics (Vice Pres-

[6]The qualitative results for TND are for $\mu = 5$.

ident and Covid-19), while nearly half of the topics that LDA and CSTM find are duplicates. DMM and GPUDMM find almost exclusively the Debate topic, leading them to have very few unique topics.

As a final display of the quality of TND and NLDA topics, we show topics from the Million Covid-19 and Election 2020 data sets for TND, NLDA, LDA, and CSTM. Figure 6.4 shows six topics, three from Million Covid-19 (top row), and three from Election 2020 (bottom row). We specifically picked topics that the other methods showed more coherence on. As we mentioned in the introduction, the flood word 'covid-19' and similar words are common in LDA, CSTM, and TND. However, these flood words are absent from the NLDA topics.

Despite the appearance of a flood word in TND's Covid-19 topics, TND and NLDA's quality is apparent in both data sets. In the Election 2020 topic set, TND and NLDA are particularly effective compared to LDA, which contains far more noise than in the Million Covid-19 topic set. CSTM fails to separate noise from content in most topics in these domain specific data sets.

In the Million Covid-19 and Election 2020 data sets, TND and NLDA are particularly effective, finding strong topics for each depicted in Figure 6.4. NLDA, in some cases, is more coherent than TND. LDA and CSTM are less effective, and each fails to find a strong topic for at least one selected topic in each of the data sets. LDA and CSTM are capable of finding coherent topics, as they do in the Testing/Symptoms, Vaccine, and Climate Change (only LDA in this case) topics, but due to noise, other topics miss the mark.

Masks/Social Distancing					Testing/Symptoms				
LDA	**CSTM**	**TND**	**NLDA μ=10**	**NLDA**	**LDA**	**CSTM**	**TND**	**NLDA μ=10**	**NLDA**
social	covid19	fight²	mask²	mask²	positive	covid19	care	test²	test³
coronavirus	mask²	covid	spread	spread	test⁴	positive	uk	symptoms	minister
china	spread	lets	face	social	symptoms	tested	social	study	home
video	help	lives	wear²	face	results	hospital	covid	sarscov2	free
safe	wear	mask	protect	protect	sarscov2	coronavirus	test	disease	state
stay	protect	save	social	wear²	friday	minister	staff	results	big
city	coronavirus	message	stay	stop	monday	anyone	nhs	big	result²
home	wearing	line	safe	stay	free	covid	distancing	infection²	infected
distancing	covid	wear	stop	prevent	died	admitted	sign	heart	negative
wuhan	deaths	staysafe	distancing	immunity	infection	say	continue	found	admitted
Vaccine					**Climate Change**				
LDA	**CSTM**	**TND**	**NLDA μ=10**	**NLDA**	**LDA**	**CSTM**	**TND**	**NLDA μ=5**	**NLDA**
vaccine²	covid19	covid	vaccine	vaccine²	demdebate	demdebate	change	make	warren²
russia	vaccine	covidupdates	treatment	study	change	sanders²	ive	change	change
worlds	coronavirus	usa	russia	sarscov2	climate	biden²	climate	climate	climate
trials	russia	cdc	trials	disease	economy	emocraticdeb:	medicare	tonight	shes
trial	first	thing	worlds	early	work	warren	demdebate	watch	feel
clinical	covid	vaccine	effective	treatment	jobs	pete	home	people	stage
effective	trials	covidusa	trial	russia	yang²	kylekulinski	message	crisis	folks
scientists	breaking	cnn	research	trial²	national	debate	voter	plan	senator
event	says	research	event	app	crisis	lemconventio	twitter	andrewyang	close
company	died	cure	clinical	clinical	plan	people	word	hard	cmclymer
Mail-in Voting					**Healthcare**				
LDA	**CSTM**	**TND**	**NLDA μ=5**	**NLDA**	**LDA**	**CSTM**	**TND**	**NLDA μ=5**	**NLDA**
trump2020	realdonaldtrum	republican	vote²	vote²	people	taxes	care	sanders²	healthcare³
maga	vote²	put	2020election	2020election²	dont	jobs	plan	warren²	plan
kag	election	call	call	call	healthcare³	biden³	proud	healthcare²	coronavirus
voting	whether	fact	election	mail	berniesanders	coun	health	klobuchar²	public
call	mail	mail	mail	service	americans	realjameswood	wait	shes	congress
wwg1wga	im	political	person	ballots²	talking	john	healthcare	give	reminder
patriots	tried	florida	start	mailin	campaign	abortions	demdebate	reminder	message
follow	absentee	system	florida	florida	working	dear	men	senator	word
mail	true	demdebate	republicans	postal	plan	raise	lost	moderator	free
florida	trump	true	ballot	poll	pay	left	usa	child	single

Figure 6.4: Topic Comparison between TND ($\mu = 0$), NLDA ($\mu = \{10, 5, 0\}$), LDA, and CSTM. Million Covid-19 Topics are the First Three Topics, and Election 2020 Topics are the Last Three. Words are Annotated with Superscript Numbers Corresponding to the Number of Variants of the Word in the Top Ten Words.

Chapter 7

Dynamic Topic-Noise Models: D-TND and D-NLDA

The lack of dynamic topic models for social media data was one of the first problems that we identified. With topic-noise models, we had created a consistent, accurate way to deal with noisy social media data. This enabled us to finally approach the temporal problem.

There have been relatively few attempts at creating a dynamic topic model over the years, but one of the first, Dynamic Topic Models (D-LDA), by Blei and Lafferty [10], provides a strong framework for temporal models. D-LDA passes the topic distribution trained in time period $t-1$ on to time period t, in order to facilitate tracking of topics over time, and to reduce the time to convergence. Other state-of-the-art topic models include the Dynamic Embedded Topic Model (D-ETM) [34], Topics over Time [101], and Topic Flow Model [27]. D-ETM is a neural topic model that projects documents and words onto an embedding space, and creating topics within the embedding space. This embedding space, similarly to D-LDA, is passed on to the next time period, allowing for topics to be tracked. Topics over Time generates topics in a similar manner to LDA, but attempts to jointly model time as a continuous probability distribution over topics. Topic Flow Model, the subject of my masters thesis work, attempts to find emerging terms and use them to build topics using related words in a word co-occurrence graph.

We aim to create a dynamic topic-noise model using Topic Noise Discriminator and NLDA as our starting points. We propose approximating topics and noise for one

time period at a time, passing the prior knowledge of the previous time period on to the next. This passing along of distributions should result in both a stronger noise distribution over time, and for more connected topics over time. We hope to see our models produce coherent topics, strongly connected over time thanks to our temporal adaptation. With the creation of a successful dynamic topic-noise model, our suite of models would be complete.

The rest of this chapter shows how we adapted topic-noise models to a temporal plane, and how they compare to other temporal models when used on social media data sets [24].

7.1 Dynamic Topic-Noise Model Approach

In this section, we define notation specific to this chapter (Section 7.1.1) and review Dynamic LDA (Section 7.1.2). We then describe how we adapt TND and NLDA to a dynamic setting to produce D-TND (Section 7.1.3) and D-NLDA (Section 7.1.4). With the goal of producing dynamic models capable of handling large social media data sets, we then propose a heuristic for improving the performance of dynamic topic models (Section 7.1.5).

7.1.1 Notation

In the case of temporal models, we use discretized time. We refer to a data set as consisting of T time periods, $\{t_0, t_1, ..., t_T\}$, where topics within a time period are constructed together.

7.1.2 Dynamic LDA

Originally proposed in 2006 by Blei and Lafferty, D-LDA, otherwise known as DTM, remains a popular choice of temporal topic model in its optimized and modern

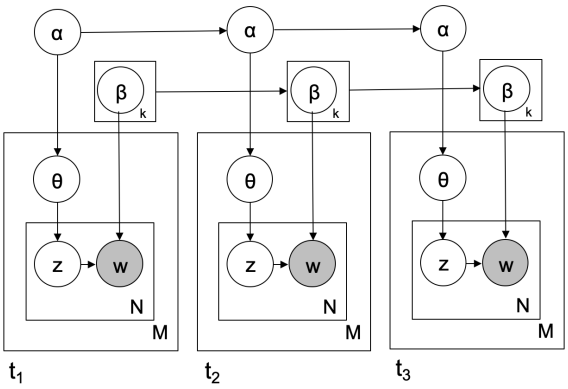


Figure 7.1: Plate Notation for D-LDA, for Three Time Periods.

form [10]. Figure 7.1 shows the plate notation for D-LDA. D-LDA defines a topic-word distribution $\beta_{t,k}$, where t is the time period, and k is the number of topics. For a document d, its document-topic distribution $\alpha_{t,d}$ is a probability distribution over β_t. When generating a word for document d on time slice t, a topic z is chosen from β_t conditioned on $\alpha_{t,d}$. The word $w_{t,d}$ is drawn from $\beta_{t,z}$.

D-LDA assumes that topics are reliant on the time period and on the topic-word distribution, which is carried over from the previous time period. In Figure 7.1, we can see how α_t and β_t are directly related to α_{t-1} and β_{t-1}, respectively. α and β are initially group Dirichlet priors (document-topic and topic-word distributions, respectively) in the first time period, but once trained, are passed to future time periods as individual priors. This allows for topics to be tracked through time periods, providing they exist in each time period. The key contribution to temporal topic models made by Blei and Lafferty is the evolution of the topic model's parameters over time periods [10]. The document-topic distribution α_t is initialized as α_{t-1}, and the topic-word distribution $\beta_{t,k}$ is initialized as $\beta_{t-1,k}$ in order to preserve topic approximations across time periods and facilitate the evolution of topics. We will draw on this temporal structure to create our dynamic topic-noise model, which incorporates a noise distribution into the model.

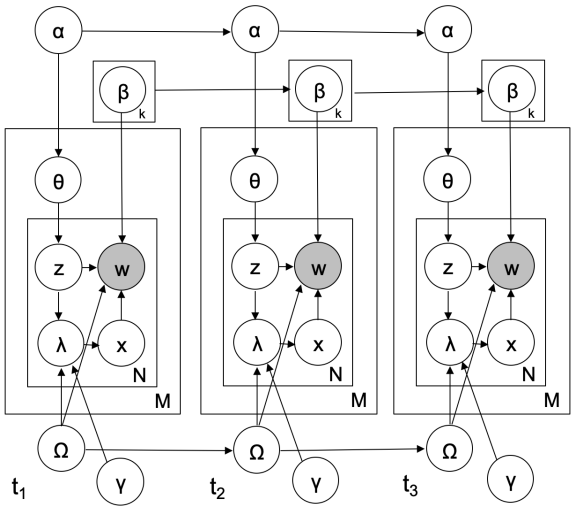


Figure 7.2: Plate Notation for D-TND, for Three Time Periods.

7.1.3 Constructing a Dynamic Topic-Noise Model

We now describe how D-TND is constructed. $\beta_{t,k}$ is the topic-word distribution for t over k topics. The document-topic distribution α_t is a probability distribution over β_t. α_t and β_t are initialized from their $t-1$ counterparts.

We define Ω_t to be the noise distribution at time t. Like α_t and β_t, Ω_t is conditioned on Ω_{t-1}. This inherently assumes that words that were noise in $t-1$ are still noise in t. While this will make it harder for noise words from $t-1$ to be included in topics, it does not make it impossible, merely less likely.

Figure 7.2 shows the plate notation for D-TND. Within a time period, observed words are chosen as noise or topic words relative to their probabilities in the chosen topic and in noise (λ), tuned by γ. This choice is indicated by the switching variable x. We describe the generative process of D-TND as follows:

For $d \in D$:

1. Draw a time period t for d.

2. Draw the number of words N for d.

3. Draw the topic distribution $\theta_{t,d}$ from the Dirichlet distribution, conditioned on α_t.

4. For each word w_i, $0 \leq i < N$:

   (a) Draw a topic z_i from the topic distribution $\theta_{t,d}$.

   (b) Draw a word from either z_i or the noise distribution Ω_t, according to λ_t, indicated by switching variable x.

   (c) If drawing from z_i, draw w_i from β_{t,z_i}.

   (d) If drawing from Ω_t, draw w_i from Ω_t.

7.1.4 Constructing Dynamic NLDA

D-TND's most versatile feature is its noise distribution, which is trained in conjunction with its topic distribution. Like TND for static models, D-TND can be easily integrated into generative temporal topic models. This makes D-TND particularly useful because researchers can use it in concert with whichever model they prefer.

Just as NLDA integrates TND's noise distribution and LDA's topic-word distribution, D-NLDA integrates D-TND's noise distribution and D-LDA's topic-word distribution. To create D-NLDA, we train a noise distribution Ω on D over all time periods $t \in T$. We generate topics on D using D-LDA, combine D-LDA's topic-word distribution β_t, k with D-TND's Ω_t to create topics for each time period. A word is removed or retained using the Beta distribution, conditioned on $\beta^i_{t,z}$ and $\Omega_{t,i}$ (Equation 7.1).

$$Beta\left(\sqrt{\beta^i_{t,z}+\gamma}, \sqrt{\Omega_{t,i}(\phi/k)}\right) \tag{7.1}$$

After the status of w_i has been determined, we follow the same guidelines as NLDA, incrementing $\Omega_{t,i}$ by one if w_i is noise, or by $\beta^i_{t,z}$ if w_i belongs to z. This ensures that, *for time period* t *only*, w_i has a high chance of not being put in another topic if it already belongs to one. As Ω has already been computed for all $t \in T$, this does not affect the status of w_i in future time periods.

7.1.5 A Preprocessing Heuristic to Improve Topic Model Performance

As we mentioned in Section 1, topic models are often too slow to infer topics on large data sets in a temporal setting. The original D-LDA [10], D-ETM [34], and ToT [101] only show results on data sets of tens of thousands of documents. In order to facilitate better scaling for topic models, we propose reducing the vocabulary size of data sets.

The idea of reducing the size of vocabulary is not new, but we formalize it here for use in dynamic topic models. We define a *vocabulary limiting function* (VLF) to be a function that removes words from the vocabulary V, resulting in a smaller vocabulary V'. We define the frequency of a word $w_i \in V$ to be f_w. Given a threshold f_{min}, we compute the VLF as follows:

$$V' = V' \cup \{w_i\} \; \forall w_i \in V | f_{w_i} > f_{min} \tag{7.2}$$

In practice, we set f_{min} such that $|V'|$ is approximately equal to some target vocabulary size.

7.2 D-TND and D-NLDA Empirical Evaluation

In this section, we present our empirical evaluation of D-TND and D-NLDA using quantitative and qualitative approaches. We begin by describing our experimental setup, including data sets, preprocessing, and model parameters (Section 7.2.1). We then present a quantitative evaluation (Section 7.2.2), and a qualitative evaluation of our models' performance (Section 7.2.3).

7.2.1 Experiment Setup

Baseline Algorithms. In our experiments, we tested against four state-of-the-art temporal topic models: D-LDA [10], ToT [101], TFM [27], and D-ETM [34]. They are each described in Section 2.

Data Sets. In our analysis, we use two Twitter data sets. The first data set contains posts about the 2020 United States Presidential Election from August 1 to November 14. We will refer to this data set as *Election 2020*. We collected these documents using hashtags related to the election via the Twitter Streaming API, and randomly sampled 200,000 posts per week, for a total of three million tweets. The second data set, referred to as *Covid-19*, contains posts about the Covid-19 pandemic, collected between March 2020 and February 2021, for a total of twelve months. Like in the Election 2020 data set, Covid-19 was collected using hashtags related to the pandemic via the Twitter Streaming API, and we randomly sample 200,000 tweets per time period, this time for a total of 2.4 million documents.

We sample 200,000 tweets per time period instead of using the full data set. Leaving data sets in their original form, with a large skew in data set size from time period to time period, reinforces the skews in more pronounced ways in the probability distributions, leaving effects on future time periods.

Table 7.1: Data Set Qualities for Different Size Variants of Vocabulary

		$\lvert D\rvert$	$\lvert V\rvert$	$\lvert V\rvert/t$
Covid-19	Large	2,400,103	1,041,552	172,116
	Medium	2,326,370	28,198	10,483
	Small	2,292,266	13,890	5,645
Election 2020	Large	3,000,042	648,193	96,671
	Medium	2,836,549	33,391	9,484
	Small	2,800,209	18,010	5,153

$|V|/t$ is Average Vocabulary Size within a Time Period.

In order to assess the scalability of models, we created three versions of each data set using a vocabulary limiting function. The first, the large version, was the full vocabulary, ($f_{min} = 0$). For the second version, the medium-size data sets, we set f_{min} such that $|V'| \approx 10,000$ for each time period.[1] For the final version, the small-size data sets, f_{min} was set such that $|V'| \approx 5,000$ for each time period.[2] Figure 7.1 shows the exact effects of the vocabulary limiting function for each data set. On the far right, we can see the average vocabulary size per time period is approximately 10,000 for the medium-size data sets, and 5,000 for the small-size data sets. This reduction relates to a large decrease in the total vocabulary size of the data sets, from over one million down to 28,000 and 13,000 in Covid-19, and from 648,000 down to 33,000 and 18,000 in Election 2020. While there is a significant reduction in vocabulary, the number of documents remains high. In Covid-19, just over 100,000 documents, or about 4%, are lost, while in Election 2020, about 200,000 documents, or about 6.67% are lost. As we will see, this loss in documents has very little effect on the quality of topics.

Text Preprocessing. Text processing can have a positive impact on topic model performance [23]. For our data sets, we tokenize on whitespace, remove lowercase

[1] $f_{min} = 15, 20$ in Election2020 and Covid-19 for the medium-size data sets.
[2] $f_{min} = 40, 50$ for Election2020 and Covid-19 for the small-size data sets.

text, remove URLs, punctuation (including hashtags), and stopwords. We also remove deleted posts and user tags.

Model Parameters. We conduct a sensitivity analysis for D-TND and D-NLDA, testing each model with an array of different parameter settings. Due to space limitations, we present the results for the best-performing settings. For D-TND, we found the best parameter settings to be $\alpha = 1$, $\beta = 0.01$, $\gamma = 25$, and $k = 30$. The best settings for D-NLDA were the same settings as D-TND, with $\phi = 10$. It is easy to quickly iterate through ϕ values, since the filtering of noise is the fastest part of the model.[3] For D-LDA, the best parameter settings were $\alpha = 1$, $\beta = 0.01$, and $k = 30$. For TFM, the best parameter settings we found were $\alpha = 5$, $\beta = 10$, $\gamma = 10$, $\delta = 1.5$, and $depth = 3$. For D-ETM and ToT, the parameters suggested in the papers were used, with $k = 30$ to match the parameters of the other models.

7.2.2 Quantitative Analysis

Evaluation Metrics. In our experiments, we use topic coherence, diversity, and quality (the product of coherence and diversity). We also care about efficiency in large temporal data sets. To measure efficiency, we use *time per iteration*. Temporal models must be able to detect topics quickly if we wish to use them with large data sets and in close to real time. An iteration refers to one pass over the entire data set.[4] In the case of models that perform iterations on one time period at a time, the average time per iteration is the sum of average times per iteration on each time period. For models that perform an iteration over all time periods at once, the average

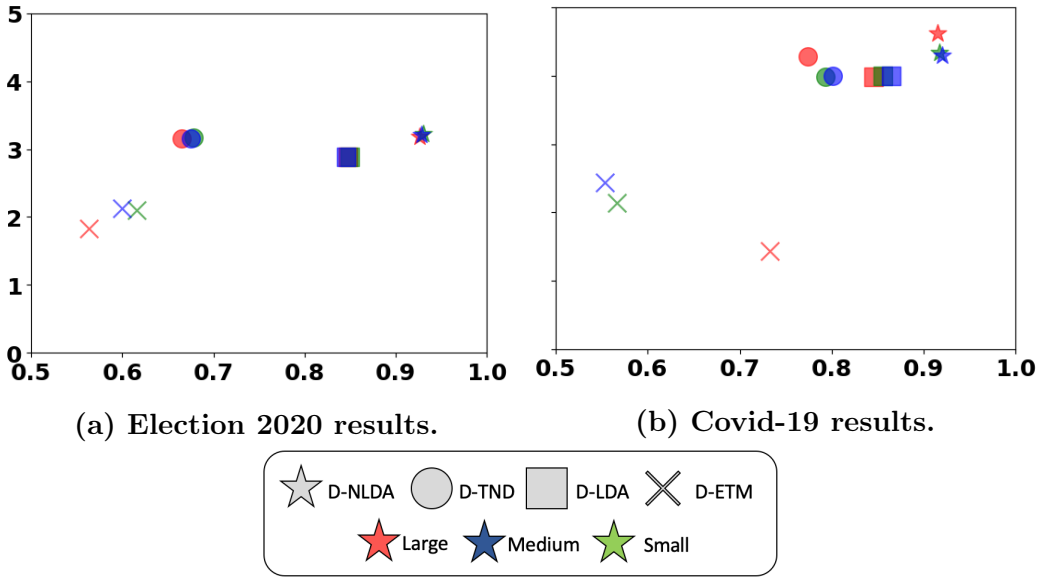
[3] Parameters for sensitivity analysis across our models: $k = \{10, 20, 30, 50, 100\}$, $\alpha, \beta = \{0.01, 0.1, 1.0\}$, $\gamma = \{0, 16, 25, 36\}$, $\phi = \{5, 10, 15, 20, 25, 30\}$

[4] In the case of TFM, which models in one pass, the time per iteration was the total time of the model.

time per iteration is straightforward. This approach makes time per iteration directly comparable across models that perform iterations differently.

Efficiency. An important aspect of temporal topic models, especially for large data sets, is efficiency. A model's performance can only be judged if the model can finish in a reasonable amount of time. Topic Flow Model (TFM) [27], Topics over Time (ToT) [101], and D-ETM [34] are three such models whose efficiency negatively affects their judgment. We narrowed down TFM parameter combinations to find the model with the most-connected graph structure that could still finish running within three days, and even then the model did not find topics with more than a single word. Due to the single word topics, its topic coherence score was zero for all data sets, meaning it was unable to create meaningful topics. By definition, its diversity is always perfect. ToT did not scale to the size of our data sets. It was allowed to run for three days, and at the end had not completed an iteration for either data set. As such, we do not include TFM and ToT in the rest of the analysis.

D-ETM is implemented using PyTorch, a highly optimized Python framework for neural networks [73]. In the documentation, 1000 iterations is suggested for training. However, as we will find in our analysis of model time per iteration, a single iteration of D-ETM took eight hours when given the exact same computational resources as the other models in the study. Extrapolating from the time for a single iteration, it would take 333 days to complete on the machine that we used to test all of our models. In order to allow for a comparison of D-ETM to other models, we chose to evaluate D-ETM on topics detected after ten iterations. Reducing the requirement to ten iterations allowed D-ETM to run on the small- and medium-size versions of both data sets, but still ran out of memory on the large data sets. The final runtime of D-ETM on the small- and medium-size data sets was between three and ten times as



(a) Election 2020 results. (b) Covid-19 results.

Figure 7.3: Coherence (y-axis) and Diversity (x-axis) Plot for Election2020 and Covid-19 Data Sets.

long as D-NLDA, which we ran for 500 iterations.[5] We successfully ran D-ETM on the large-size data sets for five iterations each. These are the results shown in this dissertation.

Coherence and Diversity. This section focuses on the quality of the models. Figure 7.3 plots the mean coherence and diversity score for D-TND (circles) and D-NLDA (stars) alongside D-LDA (squares) and D-ETM (X) for each data set. Coherence is plotted on the Y-axis, and diversity is plotted on the X-axis. The results for the large-size data set are colored red, the medium-size blue, and the small-size green. The closer to the top-right corner of the plot a model is, the better. D-NLDA performs the best out of any model on both data sets. In the Election 2020 data set, there's very little difference in D-NLDA's performance across the different-size data sets. D-NLDA has a slightly higher (1.5%) coherence than D-TND, but a 25% higher diversity score. Compared to D-LDA, its diversity is 7% higher, and its coherence

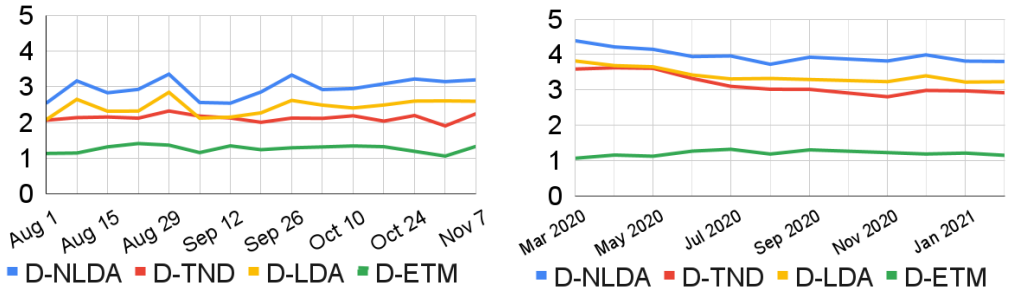
[5]D-TND and D-LDA were also run for 500 iterations.

is 8% higher. In the Covid-19 data set, D-TND is close behind D-LDA is terms of diversity, and has higher coherence on the large-size data set. D-NLDA once again is the best model on all data set sizes, beating D-TND by 0.35 in coherence and 14% in diversity, and beating D-LDA by 0.64 in coherence and 7% in diversity. D-NLDA's boost in topic coherence and diversity over D-LDA comes from the integration of D-TND's noise distribution, and indicates that D-NLDA is more successful at reducing noise in topics. In both data sets, D-ETM suffers from low coherence and diversity, never coming within 50% of the coherence score of D-NLDA. This is likely due to its inability to scale to the larger data sets, and its inability to finish enough iterations to detect high-quality topics.

One takeaway from Figure 7.3 is that the vocabulary sizes and resultant slight reductions in data set size have little to no effect on the overall quality of topics in any of the models. In fact, in the case of D-LDA on the Election 2020 data set, there is no discernible difference, and in the Covid-19 data set having a smaller vocabulary increases its diversity. D-TND and D-NLDA gain a small increase in coherence (7%) on the larger Covid-19 data set, but are almost completely unaffected in the Election 2020 data set. As such, we will focus on the medium-size data sets for the rest of our evaluation, given that it was the largest data set on which D-ETM was able to finish ten iterations within three days.

In order to understand how models perform over time in relation to one another, we plot topic quality, the product of the coherence and diversity scores, for each model in each time period in Figure 7.4. Plotting topic quality over time highlights the similarity of D-NLDA and D-LDA, but also highlights the clear improvement offered by the addition of D-TND's noise distribution.

Time Performance Comparison. We compare the time per iteration of D-TND, D-NLDA, D-LDA, and D-ETM on the Election2020 and Covid-19 data sets.



(a) Election 2020 results over time. **(b) Covid-19 results over time.**

Figure 7.4: Topic Quality Plot for Election 2020 and Covid-19 Medium-size Data Sets.

Table 7.2: Time per Iteration on each Data Set

Model	Covid-19			Election 2020		
	Large	Medium	Small	Large	Medium	Small
D-TND	19.0 s	18.2 s	14.6 s	21.9 s	15.2 s	13.8 s
D-LDA	**1.5 s**	**1.4 s**	**1.32 s**	**2.0 s**	**1.2 s**	**0.9 s**
D-NLDA	20.6 s	19.7 s	16.0 s	23.6 s	16.4 s	14.8 s
D-ETM	480 m	117 m	87 m	360 m	177 m	138 m

(s=seconds, m=minutes)

For all of our experiments, models were run on a machine with twelve 2.2GHz virtual cores, with 50GB of memory. D-TND, D-NLDA, D-LDA, and D-ETM all take advantage of parallelization or multi-threading. Table 7.2 shows the time per iteration for each model. As we can see, D-LDA is the most efficient model. Because it is only computing a topic distribution and not a noise model as well, this result is not surprising. D-TND and D-NLDA are the next most efficient models and are comparable to each other. However, the real comparison here is between our models and D-ETM. Both models are based on highly optimized code bases designed for efficiency (Mallet for D-TND [60], PyTorch for D-ETM [73]). Despite both being designed with efficiency in mind, our models are between 300 and 1500 times faster

Table 7.3: Percent Judge Agreement on Covid-19 Temporal Topics

Topic	Agreed %
Vaccines	100
Lockdowns	100
Cases	100
Testing	100
Schools	100
Masks	100
Global Impact	100
Economy	100
India	80
China	100

than D-ETM, the most recent state-of-the-art temporal topic model in our study. The gap gets smaller for the smallest versions of the Covid-19 and Election 2020 data sets, suggesting that D-ETM's complexity means that it cannot scale to data sets even larger than these. We note that on the large versions of our data sets, D-ETM was never able to complete more than eight iterations, running out of memory after six iterations on the Covid-19 data set, and eight iterations on the Election 2020 data set.

7.2.3 Qualitative Analysis

Our qualitative analysis consists of a unique look at both the Covid-19 and Election 2020 data sets, and aims to display the ability of D-NLDA to track topics through time.

For the Covid-19 data set, we asked five human judges to individually label ten topics generated by D-NLDA that persisted throughout every time period. Judges were presented with the top ten most likely words per topic in each time period, and had to give a label for each topic.[6] In Table 7.3, we show the agreement between the experts, along with the agreed upon label for each topic. For all topics but one, every judge was able to positively identify the topic in agreement with the other judges.

[6]Judges were not given a pool of possible topic labels, the responses were open-ended.

March 2020	April	May	June	July	August	September	October	November
medical	vaccine	vaccine	vaccine	vaccine	vaccine	vaccine	vaccine	vaccine
workers	family	scientists	china	sarscov2	research	trials	emergency	vaccines
back	doctors	hydroxychloroquine	world	human	sarscov2	fauci	vaccines	effective
healthcare	nurses	research	virus	study	study	trial	public	trial
rate	treatment	treatment	lives	results	effective	experts	political	family
made	dying	study	black	research	scientists	find	health	results
night	heart	evidence	united	made	vaccines	clinical	data	pfizer
full	science	prevent	matter	early	immunity	trust	mental	missed
vaccine	policy	hands	human	shows	emergency	company	information	friends
action	scientists	spread	sarscov2	trials	governments	hold	control	spent
national	didnt	effective	chinese	complete	told	ready	billion	months

December	Dec (2)	January 2021	Jan. (2)	Jan. (3)	Jan. (4)	February	Feb. (2)	Feb. (3)
vaccine	vaccines	vaccine	vaccination	people	vaccines	vaccine	vaccines	vaccination
pfizer	effective	pfizer	india	video	doses	dose	countries	house
days	data	effective	minister	sick	healthcare	received	global	biden
vaccination	hospitals	rollout	working	vaccinated	workers	covidvaccine	india	president
heres	county	received	time	work	million	vaccinated	world	years
doses	california	covidvaccine	narendramodi	laurie_garrett	countries	pfizer	make	team
vaccinated	developed	started	prime	taking	county	rollout	access	past
years	christmas	receive	families	paid	iran	single	communities	economy
covidvaccine	shows	moderna	global	weeks	high	shot	safe	vaccinations
drleanawen	kids	mrna	response	america	frontline	distribution	country	mass
reach	tomorrow	dose	corona	bill	worker	leading	ensure	white

Figure 7.5: Evolution of the Vaccine Topic in the Covid-19 Medium-size Data Set (March 2020-February 2021).

In the India topic, four out of five judges agreed that the label should be related to the country India, while the fifth judge recommended the label 'Global Response.' The latter recommendation is more general, but does not reflect a radically different understanding of the topic presented. These findings indicate that D-NLDA is able to generate high quality topics that humans are easily able to comprehend.

We highlight this ability with a deeper look at the evolution of the vaccine topic through time periods, seeing it evolve and grow. Figure 7.5 shows the evolution of the vaccine topic in the medium-size version of the data set, from March 2020 to February 2021. Words highlighted green appeared in the topic in the previous time period, and words highlighted yellow appeared in the topic in any previous time period. The topic starts out as a wish for a vaccine, with concern for healthcare workers, evolves into a reality in the middle of 2020 and goes through drug trials, and finally is approved in late 2020 and rolled out at the beginning of 2021. As we can see, the number of topics

in Figure 7.5 is greater than one for December 2020, January 2021, and February 2021. We highlight all topics that included a variant of the word 'vaccine' in their top ten most likely words.

D-NLDA allows us to see a very detailed evolution of the Vaccine topic that contains limited noise throughout the entirety of the year encompassed by the data set. The adaptation of a topic-noise model to a temporal setting results in a highly coherent and diverse set of topics, and allows researchers and observers alike to quickly understand the evolution of topics that they care about.

In the Election 2020 data set, we show the ability of D-NLDA to accurately track multiple relevant topics through time. We use the medium-size data set for this analysis. To produce topic labels for this data set, we relied on a manually-generated topic set, curated by political scientists who closely studied the 2020 Election on social media platforms [88]. Figure 7.6 shows how the proportions of selected topics change throughout the election. New topics emerge and disappear throughout the campaign. We can see the large impact of the party conventions in time periods two and three (late August 2020), and how quickly talk about conventions ceases after they are over. The same happens with topics about Presidential and Vice Presidential Debates in time periods eight to ten (early October).

The Conventions and Debates topics represent bursty topics, which appear out of nowhere and disappear quickly as attention turns away from them. These bursty topics could be missed or muddled with other topics by static models run over the whole time period.

This topic flow visualization highlights the ability of a dynamic topic-noise model like D-TND or D-NLDA to produce highly relevant and easily understandable topics with a temporal aspect. The ability to understand how topics evolve over an election

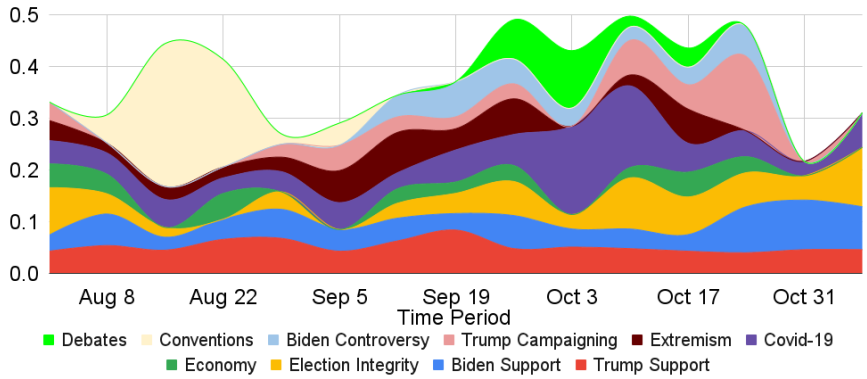


Figure 7.6: Election 2020 Topic Proportions (y) over Time Periods (x), for Selected Topics.

is important both for voters who want to be informed, and for campaigns who want to stay relevant in a world where topics of discussion change too fast to track.

Chapter 8

Guided Topic Model

A common complaint of researchers in social and political sciences is that topic sets are incomplete. As experts in their field, they often know what some of the most salient topics in their domain are, and seeing one or more of those topics missing from the results of a state-of-the-art topic model does not instill trust in the model. Noise is often a contributing factor to researchers not correctly identifying topics, and is almost always a delaying factor in the understanding of a topic set.

Armed with our topic-noise model to deal with the noise filtering, we set out to incorporate the prior knowledge of researchers into a topic model. Our model, the Guided Topic Model (GTM) [25], takes user guidance in the form of seed topics, and generates a set of semi-supervised topics based on those seeds alongside unsupervised topics. This mix of semi-supervised and unsupervised topics both satisfies experts' concerns about completeness of topic sets and also shows them new topics that they may have missed. GTM allows for users to provide feedback, in the form of updated seed topics, to iteratively hone in on a higher quality, more complete topic set.

A few semi-supervised topic models have been proposed to allow for users to provide seed topics or similar guidance for topics. Correlation Explanation Topic Model (CorEx) attempts to group words into topics based on their correlation to seed words [37]. Category-Name Guided Text Embedding Topic Model (CatE) attempts to group words into topics based on their distance to seed words in a word embedding space [61]. Guided LDA (GLDA) allows for users to provide seed words to a generative

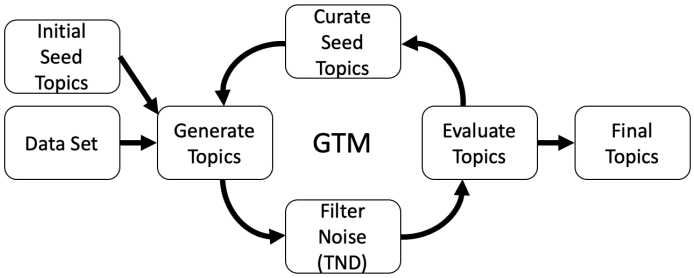


Figure 8.1: GTM Flow Chart

model based on LDA by modeling two different word distributions [43]. The first distribution is an unsupervised distribution, and the latter a supervised distribution capable of generating only seed words. Documents are generated as a mix of the two distributions. The rest of this chapter details GTM and our empirical analysis of it compared to these state-of-the-art semi-supervised topic models.

8.1 Guided Topic Model Approach

In this section, we describe the Guided Topic Model (GTM) in detail, beginning with notation, followed by the model itself.

8.1.1 Notation

A topic set Z can be initialized using a set of seed topics S. A seed topic s is identical in nature to a topic z, except that it is predefined by the user as opposed to generated by a topic model. A seed topic can also include phrases in the form of ngrams if the user believes they will be more informative.

Algorithm 2 Guided Topic Model (GTM)

INPUT: D, S, k
OUTPUT: Topic Set Z
repeat
    Z = generate_topics(k, S)
    Z = filter_noise(Z)
    done = evaluate_topics(Z) [Human]
    if !*done* **then**
        S = curate_seed_topics(Z, S) [Human]
    end if
until *done*
return Z

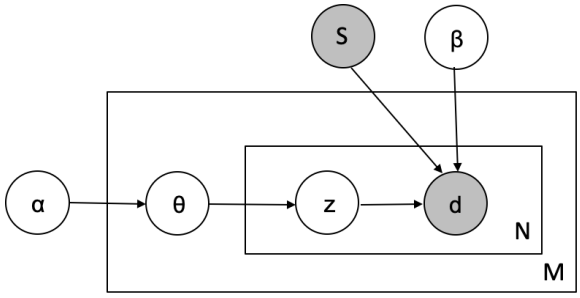


Figure 8.2: Plate Notation for GTM

8.1.2 Guided Topic Model (GTM)

Figure 8.1 shows the flow chart of how topics are produced in Guided Topic Model (GTM). At the heart of GTM is a generative semi-supervised topic model. The data set and initial seed topics are input into the generative model, and once topics have been generated, noise is filtered out using a Topic Noise Discriminator [22] trained on the data set. After, the researcher evaluates the topics, if the research is satisfied

with the output, he/she is done. However, the researcher can also adjust or augment the seed topics and repeat the process until satisfied.

Generating Topics with Guidance. The generative process of GTM differs from an unsupervised model in one clear aspect: it is given a set of seed topics as prior knowledge to guide the model. GTM also differs from traditional supervised models in one clear aspect: there are no labeled data samples, which in the case of topic modeling, would come in the form of labeled documents. All that is provided is guidance in the form of seed topics. While we could use the seed topics to label documents as belonging to one topic or another from the start, we do not view this as advantageous. Given that the size of the seed topics are expected to be very small compared to the size of the vocabulary, we would be attaching labels to documents with incomplete knowledge, and risk mislabeling documents from the outset, leading to poor topics. Here we detail our approach to using, but not overusing, the guidance provided to the model to generate highly interpretable topics.

Figure 8.2 shows the plate notation for GTM. As we can see, in GTM seed topics are injected into the generative process at the word level, as opposed to the topic or document level. More specifically, the high level generative process of GTM is as follows. However, when topics are drawn, if the topic is a seed topic, seed words are given preference.

For $d \in D$:

1. Draw the number of words N for d.
2. Draw the topic distribution θ from the Dirichlet distribution, conditioned on the parameter α.
3. For each word w_i, $0 \leq i < N$:

(a) Draw a topic z_i from θ.

(b) If seed topic s_i exists:

    i. Draw a word w_i based on the probability of w_i given z_i and S and conditioned on the parameter β.

(c) Otherwise:

    i. Draw a word w_i based on the probability of w_i given the topic z_i and conditioned on the parameter β.

Model Initialization. When initializing the model, the generative process of many unsupervised models randomly assigns a topic to each document, and then randomly assigns a topic to each word in the document. We do the same for each document and for each word that is not a seed word. However, for seed words, we have a higher probability of them remaining in the correct seed topic. We assume here that $k \geq |S|$, so seed topic i corresponds to topic i in the model.

Sampling with Guidance. Figure 8.3 shows the differences between traditional Gibbs sampling, oversampling, GPUDMM sampling, and our proposed method, GPU seed word sampling. Traditionally, LDA inspired models use Gibbs sampling, which in each iteration draws a word (a ball in Figure 8.3), and with respect to the topic distribution of the observed document, reassigns the word to a topic (replaces the ball in Figure 8.3). Gibbs sampling works well in many regards, especially in unsupervised models where there is no prior knowledge about the value of specific words. However, in the case of Guided Topic Model, we already have a list of very important words and phrases in the form of seed topics.

In Section 2, we described the Generalized Polya Urn (GPU) sampling scheme devised by Mimno et al. [65], and the GPUDMM model proposed by Li et al. [54]. Using the GPU sampling proposed by Mimno, we can oversample words (specifically

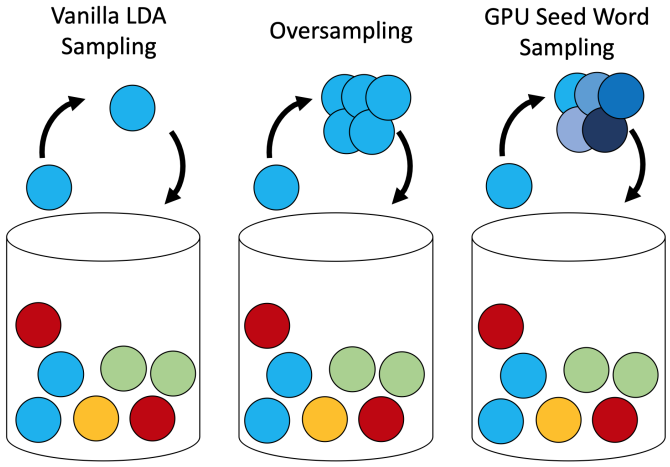


Figure 8.3: Sampling Schemes for Gibbs Sampling (LDA), Oversampling, Embedding Sampling (GPUDMM), and GPU Seed Word Sampling.

seed words) with low frequency in the data set to increase their probability in their corresponding seed topics. Figure 8.3 shows the oversampling approach in the middle, where one ball is drawn and five are replaced in its stead. In GPUDMM, the authors, instead of oversampling the observed word, used the observed word to find related words in an embedding space, and sampled each of the related words as well as the observed word. Figure 8.3 shows the embedded sampling scheme, where one ball is drawn and it is replaced alongside related words from the embedding space.

Instead of an embedding space containing related words, we are given as guidance something much more valuable. Seed topics, assembled by someone with prior knowledge of the domain, are direct confirmation of word relatedness, as opposed to the implicit relatedness given by embedding spaces. We do not need to use embedding spaces here to improve topic coherence because we already have highly related sets of seed words. Also, embedding spaces are made to be more generic, providing word relations that are most common for the word as opposed to relationships that are domain specific. For example, a general embedding space would not put the words *email* and *Clinton* close to each other. However, a researcher studying the 2016 US

Presidential election would. We call our sampling algorithm *GPU seed word sampling*. In GPU seed word sampling, when a non-seed word is observed, Gibbs sampling is performed, replacing it with a single copy of itself. However, when a seed word is observed, its whole seed topic is sampled and placed in the correct topic. Each seed word is oversampled by the product of a global factor γ and an individual factor δ_i, a word's inverse document frequency. Figure 8.3 shows GPU seed word sampling on the right, where the seed word is sampled alongside multiple copies of its fellow words from the same seed topic. By performing GPU seed word sampling on the seed topic words, we are able to maximize the probability of each seed topic regardless of the frequency of the topic in the data. This allows us to generate more complete topics for all seed topics.

In other approaches to semi-supervised topic modeling, such as CatE and CorEx, the seed words are automatically added to the correct topics. Why do we bother sampling these words instead of just adding them directly to the topic? The answer to this is simple. Oversampling the seed words is the generative equivalent of automatically adding the seed words to the correct topics. However, the point of oversampling these seed words is not to ensure their existence in the topics, it is to promote the existence of highly related non-seed words in those same topics.

8.1.3 Filtering Noise

GTM is designed with social media in mind. A common source of frustration for those using topic models on social media is noise. To filter noise from GTM, we propose using it in an ensemble with the Topic Noise Discriminator (TND), as detailed by Churchill and Singh [22]. TND works by modeling two distributions, a topic-word and noise distribution, simultaneously. When generating a document, a word can be drawn either from the topic distribution of the document, or from noise. This

Table 8.1: Data Set Sizes

Data Set	# Docs	Vocab
Survey School	2,697	2,781
Gun Violence	145,602	86,913
Covid-19	620,297	432,555
Election 2020	1,226,369	345,603
BLM	1,300,340	397,728

approach more accurately captures the random nature of noise in social media data, and produces a robust noise distribution that is implicitly conditioned on the topic-word distribution. In Section 2 we noted that DF-LDA was extended to allow for certain words to be excluded from topics altogether [48]. The advantage of using TND is that noise removal is automatic, with no user input required.

8.2 GTM Empirical Evaluation

In this section, we present our empirical evaluation of Guided Topic Model (GTM). We begin by describing our experimental setup. We then present our quantitative and qualitative evaluations.

8.2.1 Experiment Setup

Baseline Models. We compare our model to GLDA [43], CorEx [37], and CatE [61] since they are semi-supervised at the word-level, and begin with seed words, similar to GTM.[1] We also compare to unsupervised models to understand the ability of GTM to capture topics that might otherwise be overlooked or incoherent. With this in mind, we compare GTM to the following state-of-the-art topic models: Latent

[1]We do not compare to DF-LDA [48], because it does not scale to larger data sets. The inclusion of more complex expression logic has a large impact on the efficiency of the algorithm.

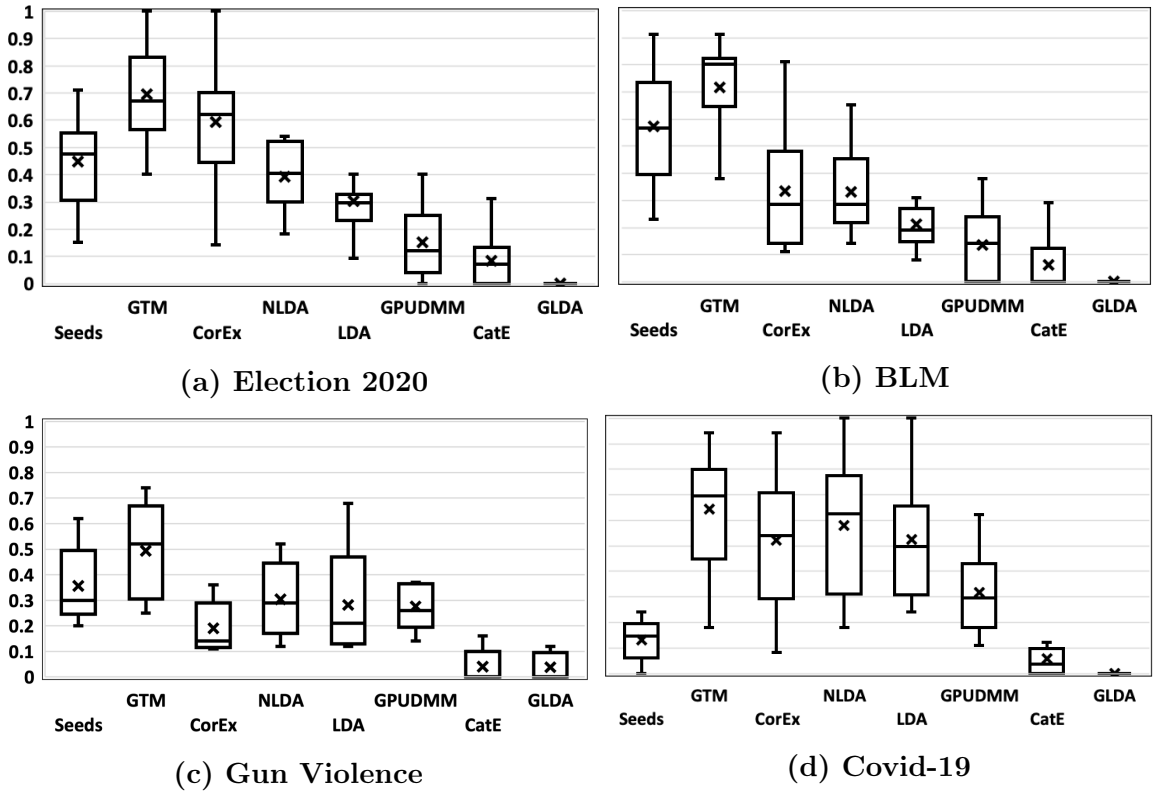
Dirichlet Allocation (LDA) [12],[2] Generalized Polya Urn Dirichlet Multinomial Mixture (GPUDMM) [54], and Noiseless Latent Dirichlet Allocation (NLDA) [22]. We include LDA because of its ubiquity and consistent performance across many types of data, GPUDMM because of its theoretical contributions to GTM and its uses in social media data, and NLDA because it filters noise and, therefore, is the current state-of-the-art for unsupervised topic modeling of social media data.

Data Sets. In this section, we evaluate model performance on four unique Twitter data sets collected using the Twitter API, and an open-ended response survey data set. All data sets are English-language.

Each of the Twitter data sets maps to domains that have recently been in the public eye. The first contains tweets about the 2020 United States Presidential Election (Election 2020). We use a data set that was collected between August 15 and November 15, 2020 using hashtags about the election, e.g. #biden and #trump. The second contains tweets about the Covid-19 Pandemic (Covid-19), collected between April 1 and December 31, 2020. This data set was collected using hashtags related to Covid-19, e.g. #coronavirus and #covid. The third contains tweets about the BlackLivesMatter (BLM) movement, collected between May 15 and July 15, 2020 using the hashtag #BlackLivesMatter. We focus on this one month because it includes when George Floyd was killed and the protests that followed. The last contains tweets about gun violence, collected in 2017, a period with multiple mass shootings. Again we use keywords and hashtags related to conversation about gun violence and gun rights. The Twitter data sets range from over 145,000 tweets to 1.3 million.

We also have survey data that included multiple open-ended survey responses. The survey consists of a probability-based web panel of 9,544 U.S. adults recruited randomly based on a nationally representative ABS (Address Based Sample) probability

[2]Specifically the MALLET implementation of LDA [60]



(a) Election 2020

(b) BLM

(c) Gun Violence

(d) Covid-19

Figure 8.4: Topic Recall Box Plots on Twitter Data Sets. x denotes Mean Recall.

design [29]. Responses were collected between March 11 and June 13, 2021. The question that we focus on for our experiments asked about challenges related to children learning from home during the pandemic. For this question, there were approximately 2,700 responses. Table 8.1 contains the sizes of each data set and vocabulary after preprocessing was applied.

Data Preprocessing. Churchill and Singh [23] show that text preprocessing can significantly improve topic model performance. We use the following elementary pattern-based preprocessing for all the data sets: lowercase, remove stopwords, remove punctuation. For the Twitter data sets, we also remove URLs, deleted posts and user tags.

Model Parameters. We conduct a sensitivity analysis for GTM, testing a range of parameter settings and determining which parameters were the most sensitive. Because of space limitations, we only present the results for the best performing settings. We determined the best parameter values by comparing topic diversity, recall, and entropy on the smallest data set, the survey data. For GTM, the best parameters were $k = 5|S|$, $\alpha = 0.1$, $\beta = 0.01$, $\gamma = 1$. We experimented with $k = x|S|$, where $x \in (1, 10)$, and $\gamma = \{1, 2, 5, 10, 20, 50, 100\}$. For TND, we used the suggested parameters provided by the authors [22], with $k = 5|S|$ and $\mu = 0$. For ensembling GTM and TND to filter noise, we set $\phi = 10$. For LDA, we found $\alpha = 0.1$, $\beta = 0.01$, and for GPUDMM, we found $\alpha = 0.1$, $\beta = 0.1$ to be the best parameter settings. For GPUDMM, we used GloVe Twitter word embeddings [74] with 50 dimensions. For both models, we used the same k value as was used for GTM to provide the best comparability. For GLDA, CatE, and CorEx, we used our seed topics, with $k = 50$.[3]

Topic model implementations. GTM is implemented using MALLET [60], a parallelized Java implementation of generative models, including LDA. This implementation is efficient because it is parallelized and because it converges faster than other implementations of generative models. All of the models we run are for the same number of iterations on the same number of threads on the same machine (24 2.2GHz vCPUs, 40GB memory).[4]

8.2.2 Quantitative Evaluation

In order to test GTM, we asked experts for seeds topics to guide the model. We used the seed topics as the seeds for GTM, CorEx, CatE, and GLDA. We also asked for

[3]CatE was designed to be given one word per topic. We tested using different 'best' words per seed topic and found that the results were better when the entire seed topic was used.

[4]The runtime of our implementation is slower than the MALLET implementations of LDA and NLDA due to the more complicated sampling scheme required, but is still faster than CorEx, CatE, GLDA, and GPUDMM.

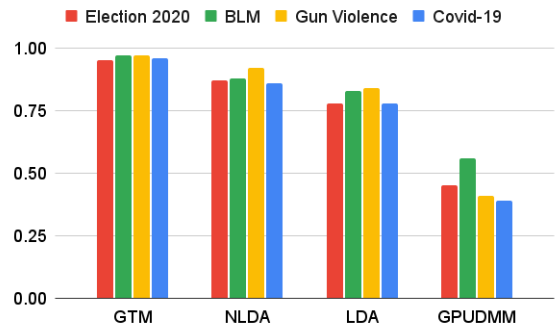


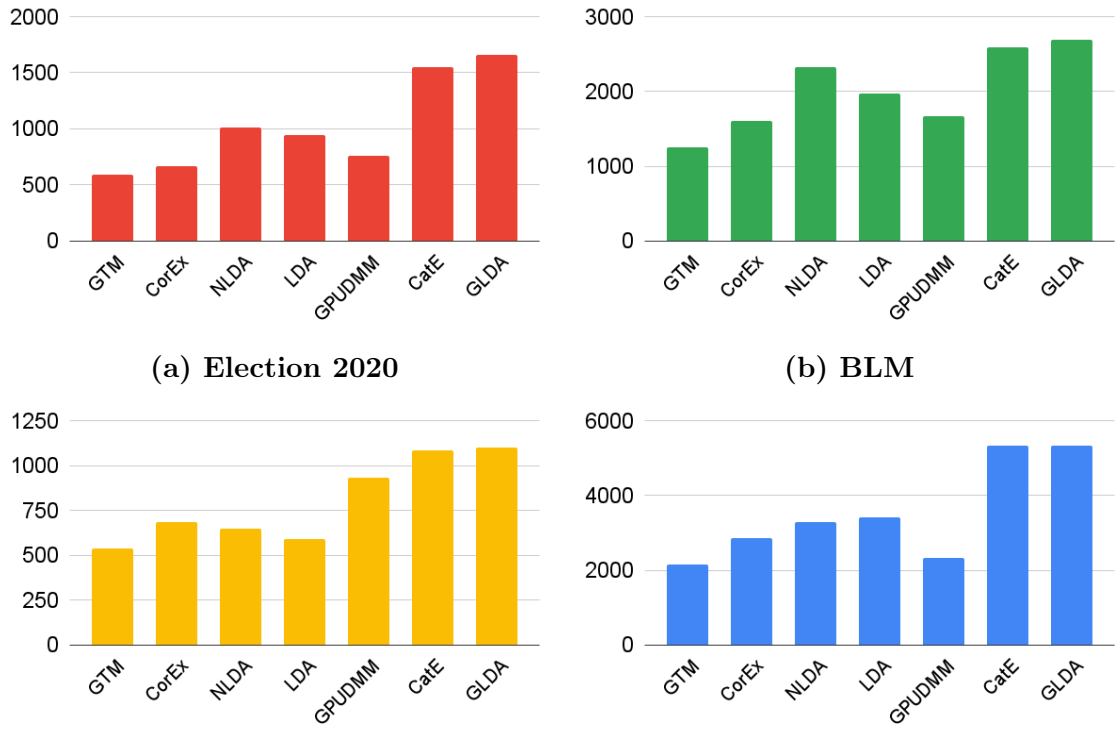
Figure 8.5: Topic Diversity Comparison between GTM and the Unsupervised Models.

extended topics, curated by the experts, for evaluation. We use these expert curated topics with the goal being to produce a set of topics as close to these expert curated topics that we will refer to as ground truth topics. To measure the ability of models to produce topics similar to the ground truth, we employ *topic recall*.

We also seek to measure the improvement offered by using GTM to augment seed topics. We had experts choose the words generated by GTM under the guidance of the seed topics that they believe truly belong to the underlying topic. Topic improvement is the percent increase in topic size after augmenting the seed topics using GTM.

Finally, we use topic diversity and entropy to judge the quality and informational value of topics.

Comparing GTM to Unsupervised Models. An important distinction between GTM and unsupervised models is its adherence to the seed topics that guide it. But how well does it adhere, and does it find unique topics for each seed topic, or does it repeat topics and lead to a situation where it's difficult to decide which approximated topic was generated by which seed topic? Furthermore, do we really need GTM to produce topics that experts expect, or can unsupervised models



(a) Election 2020 **(b) BLM**

(c) Gun Violence **(d) Covid-19**

Figure 8.6: Topic Entropy Histograms on Twitter Data Sets.

accomplish this task? To address these questions, we compute topic recall, topic entropy, and diversity.

Figure 8.4 displays a box plot of topic recall for each model on each Twitter data set. The semi-supervised models will be discussed in the next subsection. In this section, we focus on the relationship between GTM and NLDA, LDA, and GPUDMM. As we can see, in the Election 2020, BLM, and Gun Violence data sets, there is a stark contrast between GTM and the unsupervised models. The mean recall for GTM is nearly twice as high as any unsupervised model. In the Covid-19 data set, the unsupervised models performs better, with NLDA's average recall at 58%, but still not was well as GTM (64%). The unsupervised models are not able to provide the same level of recall that GTM is capable of since they are not given any guidance.

While not surprising, it is important to verify. We also pause to note that in all cases, the seeds alone do not sufficiently describe any topic. Additional meaningful words are added to every topic by GTM across all the data sets, highlighting the difficulty for researchers to identify all the meaningful words apriori.

Figure 8.6 displays the topic entropy of GTM and all of the baseline models on the Twitter data sets, where a lower score is better. As we can see, GTM is consistently the best across each data set, from 7 to 22% better than the next best model. While LDA, NLDA, and GPUDMM each perform well on one or two data sets, none perform as well or as consistently as GTM, highlighting that topic words are consistently ranked higher across topics than they are for other methods. This shows the positive impact of using semi-supervision to detect topics that domain experts care about.

Figure 8.5 shows a histogram of the topic diversity scores for GTM and each unsupervised model on each Twitter data set.[5] What we see here is that GTM scores higher in diversity than NLDA, LDA, and GPUDMM. This is a consequence of using TND [22] for noise filtering, and also a mark of the helpfulness of seed topics. NLDA also uses TND for noise filtering, and improves on the diversity of LDA by 5-10% depending on the data set, but GTM improves on NLDA's diversity by 5-10% because of the guidance of the seeds. This high diversity means that GTM is both accurate and precise, recovering more topic words than other models, but also without repeating topics. Surprisingly, considering its relative success on topic ranking, GPUDMM has very low diversity across the board, ranging from 0.39 to 0.56.

Comparing GTM to Semi-supervised Models. We now focus on comparing GTM to the semi-supervised model with similar aims. We compare to GLDA, CatE, and CorEx given that their approaches are most similar to ours. Beginning with

[5]The semi-supervised models are not pictured because CorEx and CatE, by their models' definitions, force words into only one topic. GLDA's diversity was poorer than that of GPUDMM due to noise. So we focus this analysis on the comparison to unsupervised models.

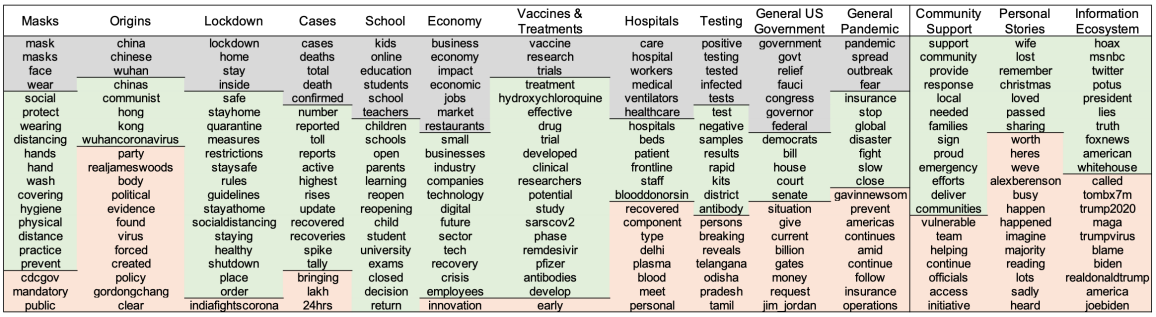


Figure 8.7: Covid-19 Twitter Topics. Top 20 words per topic. Seed words (grey shading), new topic words (green shading), and non-topic words (orange shading).

Table 8.2: Topic Improvement

	Survey	Election	Covid-19	GV	BLM
GTM	**34% (3.2)**	**176% (8.8)**	**402% (25.5)**	**114% (10.2)**	**108% (9.4)**
CorEx	8% (0.67)	120% (6.0)	220% (20.9)	69% (6.6)	10% (0.8)
CatE	15% (0.77)	82% (4.1)	47% (3.2)	38% (4.0)	29% (2.5)
GLDA	6% (0.67)	0% (0)	0% (0)	0% (0)	0% (0)

Average percent increase and average total increase (in parens) in size of topics after augmentation using GTM and CorEx.

recall in Figure 8.4, CorEx came close to GTM in the Election 2020 and Covid-19 data sets, but was closer to the recall scores of the unsupervised models in the BLM and Gun Violence data sets. GTM had better recall for all of its topics and had less spread across all of them. GTM's improvement over CorEx ranged from about 10% on the Election 2020 data set to nearly 40% on the BLM data set. CatE and GLDA performed surprisingly poorly. Judging from the topics themselves, GLDA seemed to suffer heavily from noise inundation, with all of its topics containing words that had little to do with the seed topics, but which had high frequencies. CatE performs a little bit better, but again suffers from noise problems. In the settings for CatE where we used only one seed word per topic, the amount of noise was even greater.

Returning to Figure 8.6, we now evaluate the topic entropy of the semi-supervised models (lower entropy is better). CorEx produces topics with entropy close to that

of GTM, but is beaten by GPUDMM in the Covid-19 data set and by NLDA and LDA in the Gun Violence data set. CatE and GLDA both have trouble dealing with noise words in the Twitter data sets, and for that reason have very high entropy, meaning that they are not able to accurately rank words in their optimal topics. The fact that an unsupervised model designed for social media (NLDA) is able to produce lower-entropy topics than semi-supervised models highlights how important it is to account for noise in social media data.

To further compare the semi-supervised models, we use topic improvement as shown in Table 8.2. We compare the average topic improvement of GTM, CorEx, CatE, and GLDA on each data set. The percent improvement is followed by the average number of words added per topic in parentheses. The improvement numbers do not account for completely new topics that were added. In the case of the survey data set, the improvements are much more modest because experts curated longer seed word lists than with the Twitter data sets. Even with the larger seed word list, GTM still resulted in an improvement of 34%, with an average of 3.2 words added per topic. Using CorEx and GLDA, only 0.67 words were added per topic, for an increase of 8% and 6%, respectively. The percent increase varies because the size of seed topics varies. So, if one word was added to a topic of size 4, that would relate to an increase of 25%, while adding one word to a topic of size 10 would only be a 10% increase. Having a lower average percentage increase but the same average number of words added per topic means that more words were added to larger topics than to smaller ones, which we see as having less impact on the overall topic set. In the Twitter data sets, the seed topics ranged from five to fifteen seed words per topic. In this case, the average improvement was over 100% for each data set, peaking at over 400% on the Covid-19 Twitter data set, which had an average of seven seed words per topic. CorEx and CatE did not add as many words per topic for any of the Twitter

data sets. CorEx was considerably better than CatE in every data set except for the Survey and BLM data sets. Due to their inability to deal with the noise levels of social media data, CatE and GLDA did not produce good topics on the Twitter data sets.

Using GTM as the means of augmenting and identifying new topics can help save valuable time for domain experts who want to understand their data quickly, but who already have partial knowledge of what exists. Even in the case of the Covid-19 Survey data set, where experts had already performed an extensive examination, GTM still offers improvements in topic size and quality.

8.2.3 Qualitative Evaluation

Improving Expert Understanding of Data Sets. One important aspect of GTM is its ability to create highly relative topics around the seeds while at the same time unveiling topics that experts may have missed. We demonstrate this using the Twitter Covid-19 data set. Experts identified 11 seed topics in the data. After reviewing the GTM topics, they discovered that they had actually missed three topics – community support, personal stories, and information ecosystem. Figure 8.7 shows the top 20 words for each topic guided by a seed, as well as the three new topics. Seed words are highlighted in gray on top of each column, new topic words are highlighted in green in the middle, and non-topic words are highlighted in red at the bottom. The three new topics are on the right side, separated from the seeds by a vertical line. As we can see, nearly all of the words in the first seven topics (except for the Origins topic) are seed or topic words. About half are for the rest, including the three topics discovered without the use of seeds. The large green band emphasizes the value of a guided model for improving topics of interest to researchers and also for identifying new topics.

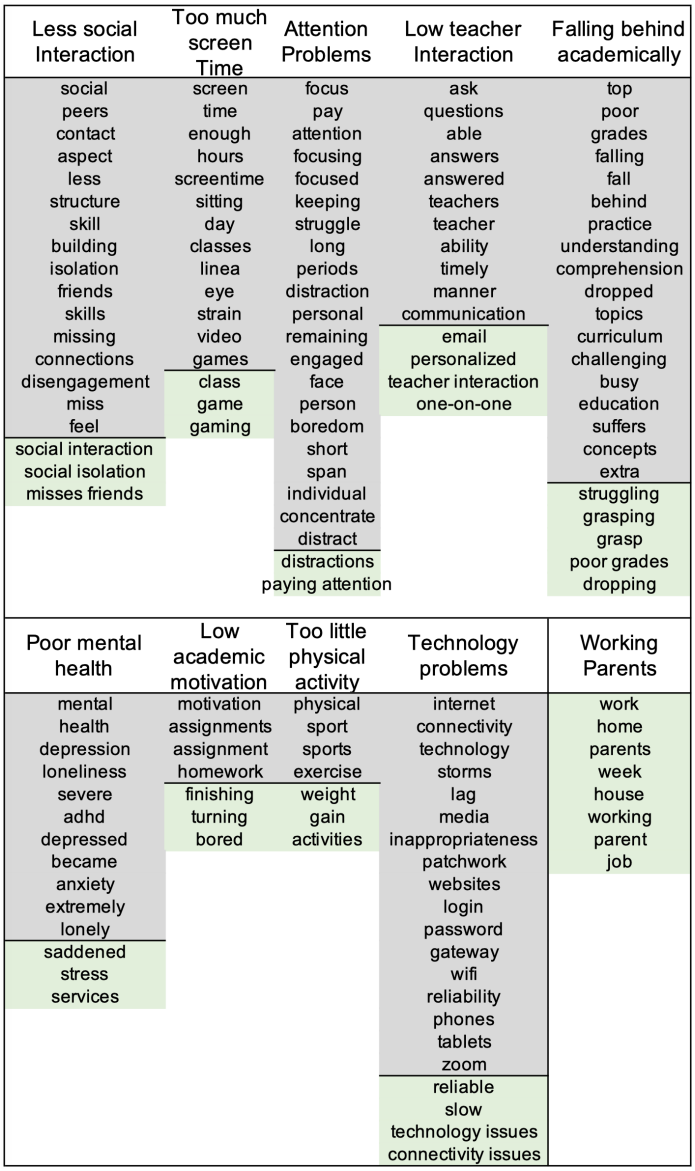


Figure 8.8: Home-schooling Survey Topics. Seed words (grey shading), new topic words(green shading).

Covid-19 Remote Schooling Challenges Survey. Figure 8.8 shows the final topics decided on by the experts after using GTM to augment their seed topics. Like Figure 8.7, the seed words are at the top of each column in gray, and the new topic words are in green. First, we can see on the right side of the bottom row that the researchers found a new topic, about working parents, in the topic set. These new words did not count toward the 34% improvement since the topic was not a seed topic, but together they are another important topic underlying the survey responses. In the other topics, we can see many highly informative words and phrases, such as 'one-on-one', 'weight', 'gain', and 'poor grades'. These added words all add to the context of their respective topics, making for higher quality, more interpretable topics.

The domain experts who curated these topics had two goals in doing so. The first was to get a more complete, descriptive set of topics to convey the main concerns and opinions of respondents to the survey. The second was to quantify those concerns and opinions by classifying responses using the topics. Because manually-curated topics consist of only a small portion of the vocabulary, we do not expect every single response to be classified under the topic set. However, we want to maximize the number that are classified, and by adding more words to each topic we hope to improve that number. The manually curated seed topics alone were able to classify 70% of the responses. Using GTM to augment the researchers' seed topics, we were able to increase that figure to 77.3%, an improvement of over 7%, and a total of nearly 200 more classified responses.

GTM and CorEx Qualitative Comparison. In our final qualitative analysis, we chose a seed topic from the Election 2020, BLM, and Gun Violence data sets and show the resulting topics produced by GTM and CorEx. Figure 8.9 shows the top 20 words for three topics as they were discovered by GTM and CorEx. The three topics are Mail-In Voting (Election 2020), Victims (BLM), and Gun Ownership (Gun

Violence). To label words as topic or non-topic words, we compared topics to expert-labeled topics. The words are ordered by topic and non-topic words, with topic words highlighted in green, and non-topic words in red.

The Election 2020 Mail-In Voting topic was found with high accuracy by both models. Each model produces topics with 15 words chosen from the top 20. The other five words in CorEx are clearly noise words, including four user handles and the generic word 'says'. On the other hand, the five words not chosen by the experts from GTM are more likely less relevant topic words that do not add enough context to be added to the final topic. Words such as 'joebidens', 'changed', and 'stake' may all have to do with voting in general, referring to one of the candidates, someone changing their mind, and the stakes at play by allowing or disallowing mail-in voting.

The differences in the BLM Victims topic are more stark. In GTM, nearly all words refer to victims of police violence. However, in CorEx, the topic seems to have been mixed together with a different topic related to signing petitions (as well as and noise words like user handles). Bowman-Williams et al. showed how, in the case of the BlackLivesMatter domain, millions of tweets were posted focusing on the victims [15], and as such a topic about victims should be easily detectable. In this case, the overlap of the underlying petition topic led to the failure of CorEx to isolate topics related to the seed words. Only one victim name is in the top 20 words. We note that LDA, NLDA, and GTM all detected a petition topic. In the case of GTM, it was a topic that was not a seeded topics. This example topic is representative of the performance of CorEx on the BLM data set as a whole. Because it relies on word correlations, correlations that cross topics can lead to muddled topics.

In the final topic here, the Gun Ownership topic, we see another interesting development. GTM and CorEx both find words relevant to the seed topics, but the topics are distinctly different. CorEx's topic refers specifically to the National Rifle Associ-

Election 2020 - Mail-In Voting		BLM - Victims		GV - Gun Ownership	
GTM	CorEx	GTM	CorEx	GTM	CorEx
ballot	ballots	george	elijahmcclain	amendment	stopthenra
early	court	georgefloyd	sign	fight	concealed
voter	voter	floyd	thread	rights	carry
fraud	ballot	breonna	petitions	owner	reciprocity
ballots	fraud	taylor	petition	regulated	hr38
state	early	tamir	byersfilms	constitution	opposeccr
mail	mail	rice	help	arms	stopccr
county	state	rayshardbrooks	tpwkhollands	2ndamendment	nras
illegal	supreme	eric	nonblack	militia	antinra
absentee	votes	ahmaud	ways	bear	arming
send	voting	arbery	signatures	carrying	credomobile
mailin	absentee	brooks	shawnwasabi	2017	bill
register	mailin	remember	kiesdaya	hate	sign
polling	electoral	rayshard	ardentlyswift	wearorange	dream
investigation	courts	garner	support	pledge	must
changed	tomfitton	martin	seconds	written	priority
entire	loudobbs	elijah	sha_elise24	society	fetus
find	sidneypowell1	class	danitycafe	awareness	senate
joebidens	jsolomonreports	mefeater	educational	racism	block
stake	says	wouldve	featuring	biases	abortion

Figure 8.9: Topic Comparison. New topic words (green shading), and non-topic words (orange shading).

ation (NRA), while the GTM topic focuses more on Second amendment rights. Both add in more general terms, in the case of CorEx, some that are not relevant to gun violence, e.g. abortion and fetus.

Chapter 9

Future Work

Improving topic models in the ways that we have has only led us to see more ways to improve and adapt them to other settings. Topic models still do not scale well to very large data sets. This can be ameliorated to some degree with parallelization, but can probably be further-improved with better code optimization and adaptation to frameworks like Apache Spark and serverless cloud computing infrastructures.

Topic-noise models have been shown to excel on Twitter data, but are so far unproven on other popular social media text data sets, like Reddit and Facebook. Some of this is due to lack of public availability, but we can certainly experiment on publicly available data like that of Reddit. Another area where topic-noise models are unproven is in languages other than English. It is an open question whether TNMs will be as effective in other languages due to the differences in how noise might be generated from language to language. It is far from certain, but we should certainly find out.

Creating static and dynamic topic-noise models has also opened our eyes to the possibility of integrating other types of probability distributions into topic models. One obvious path for such an endeavor would be the integration of a geo-spatial distribution of documents and topics. In many cases, being able to visualize social media topics over not only time, but space as well, would help pinpoint events and issues in specific areas.

Chapter 10

Conclusions

In this dissertation, we have contributed a survey paper detailing the evolution of topic models from their inception to today. We have defined noise in the context of topics, and have proposed multiple approaches to dealing with noise in social media data. While we did not quite accomplish our goal of finding the holy grail (of topic modeling), we succeeded in creating a suite of models which, taken together, meet the criteria that we outlined as necessary for a holy grail of topic modeling.

The foundation of this set of models is the new class of model, the topic-noise model, which jointly models the underlying noise and topic distributions of a data set. We use our topic-noise model, TND, as the basis for creating robust models for social media data in both static (NLDA) and dynamic settings (D-NLDA). TND is instrumental in removing noise from user-guided topics in GTM, allowing for higher quality guided topic sets and faster understanding of data sets. Most importantly, these models are all usable and accessible to those who want to use them. We publish our model, evaluation, and data preprocessing code for others to use in the pursuit of the holy grail (of topic models).[1][2][3] While they certainly benefit from faster machinery, these models can all be run on the every-day computer.

[1] textPrep code can be found here: https://github.com/GU-DataLab/topic-modeling-textPrep

[2] Topic-Noise Model code can be found here: https://github.com/GU-DataLab/topic-modeling-topic-noise-models

[3] Guided Topic Model code can be found here: https://github.com/GU-DataLab/topic-modeling-guided-topic-model

References

[1] Deepak Agarwal and Bee-Chung Chen. flda: matrix factorization through latent dirichlet allocation. In *International Conference on Web Search and Data Mining (WSDM)*, pages 91–100, 2010.

[2] Felipe Almeida and Geraldo Xexéo. Word embeddings: A survey, 2019.

[3] David Andrzejewski, Xiaojin Zhu, and Mark Craven. Incorporating domain knowledge into topic modeling via dirichlet forest priors. In *International Conference on Machine Learning (ICML)*, pages 25–32, 2009.

[4] International Publisher's Association and World Intellectual Property Organization. The global publishing industry in 2018. https://www.internationalpublishers.org/copyright-news-blog/958-wipo-and-ipa-publish-international-publishing-industry-statistics, 2020. Accessed: 04-10-20.

[5] Daniel Backenroth, Zihuai He, Krzysztof Kiryluk, Valentina Boeva, Lynn Pethukova, Ekta Khurana, Angela Christiano, Joseph D Buxbaum, and Iuliana Ionita-Laza. Fun-lda: a latent dirichlet allocation model for predicting tissue-specific functional effects of noncoding variation: methods and applications. *The American Journal of Human Genetics*, 102(5):920–942, 2018.

[6] Arindam Banerjee and Sugato Basu. Topic models over text streams: A study of batch and online unsupervised learning. In *SIAM International Conference on Data Mining (SDM)*, pages 431–436, 2007.

[7] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155, 2003.

[8] Paulo Bicalho, Marcelo Pita, Gabriel Pedrosa, Anisio Lacerda, and Gisele L Pappa. A general framework to expand short text for topic modeling. *Information Sciences*, 393: 66–81, 2017.

[9] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

[10] David M Blei and John D Lafferty. Dynamic topic models. In *International Conference on Machine Learning (ICML)*, pages 113–120, 2006.

[11] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 601–608. 2002.

[12] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 3 2003.

[13] Vincent D Blondel, Jean loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, (10), 2008.

[14] Leticia Bode, Ceren Budak, Jonathan M Ladd, Frank Newport, Josh Pasek, Lisa O Singh, Stuart N Soroka, and Michael W Traugott. *Words that matter: How the news and social media shaped the 2016 Presidential campaign*. Brookings Institution Press, 2020.

[15] Jamillah Bowman Williams, Naomi Mezey, and Lisa Singh. # blacklivesmatter—getting from contemporary social movements to structural change. *California Law Review Online*, 12, 2021.

[16] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[17] Stefan Bunk and Ralf Krestel. Welda: Enhancing topic models by incorporating local word context. In *ACM/IEEE on Joint Conference on Digital Libraries*, pages 293–302, 2018.

[18] Mario Cataldi, Luigi Di Caro, and Claudio Schifanella. Emerging topic detection on twitter based on temporal and social terms evaluation. In *KDD Workshop on Multimedia Data Mining*, 2010.

[19] Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David M. Blei. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[20] Chaitanya Chemudugunta, Padhraic Smyth, and Mark Steyvers. Modeling general and specific aspects of documents with a probabilistic topic model. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[21] Rob Churchill and Lisa Singh. Percolation-based topic modeling for tweets. In *KDD Workshop on Issues of Sentiment Discovery and Opinion Mining (WISDOM)*, 2020.

[22] Rob Churchill and Lisa Singh. Topic-noise models: Modeling topic and noise distributions in social media post collections. In *International Conference on Data Mining (ICDM)*, 2021.

[23] Rob Churchill and Lisa Singh. textprep: A text preprocessing toolkit for topic modeling on social media data. In *The DATA Conference*, 2021.

[24] Rob Churchill and Lisa Singh. Temporal topic-noise models for social media data sets. In *(Under Review)*, 2022.

[25] Rob Churchill and Lisa Singh. A guided topic model for social media data. In *(Under Review)*, 2022.

[26] Rob Churchill and Lisa Singh. The evolution of topic modeling. *(Under Review)*, XX (X):XXX, 2022.

[27] Rob Churchill, Lisa Singh, and Christo Kirov. A temporal topic model for noisy mediums. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2018.

[28] Rob Churchill, Lisa Singh, and Josh Pasek. The impact of pre-processing classes on meaningful topics from online text data. In *MIDAS Symposium*, 2018.

[29] P. Davis-Kean, R. Ryan, L. Singh, and N. Waters. Groundhog day: Homeschooling in the time of covid-19. *MOSAIC Data Brief: August 2021. Measuring Online Social Attitudes and Information Collaborative*.

[30] Henrique Ferraz de Arruda, Luciano da Fontoura Costa, and Diego R. Amancio. Topic segmentation via community detection in complex networks. *Chaos*, 26, 2016.

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

[32] Imre Derényi, Gergely Palla, and Tamás Vicsek. Clique percolation in random networks. *Physical review letters*, 94(16):160–202, 2005.

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

[34] Adji B. Dieng, Francisco J. R. Ruiz, and David M. Blei. The dynamic embedded topic model. *CoRR*, 2019.

[35] Adji B Dieng, Francisco JR Ruiz, and David M Blei. Topic modeling in embedding spaces. *arXiv preprint arXiv:1907.04907*, 2019.

[36] Chris Ding, Tao Li, and Wei Peng. On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics & Data Analysis*, 52(8):3913–3927, 2008.

[37] Ryan J Gallagher, Kyle Reing, David Kale, and Greg Ver Steeg. Anchored correlation explanation: Topic modeling with minimal domain knowledge. *Transactions of the Association for Computational Linguistics (TACL)*, 5:529–542, 2017.

[38] Matthew Hoffman, Francis R. Bach, and David M. Blei. Online learning for latent dirichlet allocation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 856–864. 2010.

[39] Thomas Hofmann. Probabilistic latent semantic indexing. In *SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, 1999.

[40] InternetLiveStats. Twitter usage statistics. http://www.internetlivestats.com/twitter-statistics/, 2019. Accessed: 2017-05-05.

[41] Hemant Ishwaran, J Sunil Rao, et al. Spike and slab variable selection: frequentist and bayesian strategies. *The Annals of Statistics*, 33(2):730–773, 2005.

[42] Tomoharu Iwata, Shinji Watanabe, Takeshi Yamada, and Naonori Ueda. Topic tracking model for analyzing consumer purchase behavior. In *International Joint Conference on Artificial Intelligence*. Citeseer, 2009.

[43] Jagadeesh Jagarlamudi, Hal Daumé III, and Raghavendra Udupa. Incorporating lexical priors into topic models. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 204–213, 2012.

[44] Hamed Jelodar, Yongli Wang, Chi Yuan, Xia Feng, Xiahui Jiang, Yanchao Li, and Liang Zhao. Latent dirichlet allocation (lda) and topic modeling: models, applications, a survey. *Multimedia Tools and Applications*, 78(11):15169–15211, 2019.

[45] Matthew L Jockers and David Mimno. Significant themes in 19th-century literature. *Poetics*, 41(6):750–769, 2013.

[46] G Kaminka et al. A joint model for sentiment-aware topic detection on social media. In *European Conference on Artificial Intelligence*, page 338, 2016.

[47] Shiva Prasad Kasiviswanathan, Prem Melville, Arindam Banerjee, and Vikas Sindhwani. Emerging topic detection using dictionary learning. In *International Conference on Information and Knowledge Management (CIKM)*, 2011.

[48] Hayato Kobayashi, Hiromi Wakaki, Tomohiro Yamasaki, and Masaru Suzuki. Topic models with logical constraints on words. In *Workshop on Robust Unsupervised and Semisupervised Methods in Natural Language Processing*, 2011.

[49] Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. Latent dirichlet allocation for tag recommendation. In *ACM Conference on Recommender Systems*, pages 61–68, 2009.

[50] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International Conference on Machine Learning (ICML)*, pages 957–966, 2015.

[51] John D Lafferty and David M Blei. Correlated topic models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 147–154, 2006.

[52] Ken Lang. 20 newsgroups dataset, 1995. URL `http://people.csail.mit.edu/jrennie/20Newsgroups/`.

[53] Jey Han Lau, David Newman, and Timothy Baldwin. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 530–539, 2014.

[54] Chenliang Li, Haoran Wang, Zhiqian Zhang, Aixin Sun, and Zongyang Ma. Topic modeling for short texts with auxiliary word embeddings. In *SIGIR Conference on Research and Development in Information Retrieval*, pages 165–174, 2016.

[55] Fangtao Li, Sheng Wang, Shenghua Liu, and Ming Zhang. Suit: A supervised user-item based topic model for sentiment analysis. In *AAAI Conference on Artificial Intelligence*, volume 28, 2014.

[56] Ximing Li, Yue Wang, Ang Zhang, Changchun Li, Jinjin Chi, and Jihong Ouyang. Filtering out the noise in short text topic modeling. *Information Sciences*, 456:83–96, 2018.

[57] Ximing Li, Jiaojiao Zhang, and Jihong Ouyang. Dirichlet multinomial mixture with variational manifold regularization: Topic modeling over short texts. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 7884–7891, 2019.

[58] Chenghua Lin and Yulan He. Joint sentiment/topic model for sentiment analysis. In *International Conference on Information and Knowledge Management (CIKM)*, pages 375–384, 2009.

[59] Qingqing Long, Yilun Jin, Guojie Song, Yi Li, and Wei Lin. Graph structural-topic neural network. In *International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 1065–1073, 2020.

[60] Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. 2002.

[61] Yu Meng, Jiaxin Huang, Guangyuan Wang, Zihan Wang, Chao Zhang, Yu Zhang, and Jiawei Han. Discriminative topic mining via category-name guided text embedding. In *The Web Conference (WWW)*, pages 2121–2132, 2020.

[62] Yishu Miao, Lei Yu, and Phil Blunsom. Neural variational inference for text processing. In *International Conference on Machine Learning (ICML)*, volume 48, pages 1727–1736, 2016.

[63] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[64] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119, 2013.

[65] David Mimno, Hanna Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. Optimizing semantic coherence in topic models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 262–272, 2011.

[66] Christopher E. Moody. Mixing dirichlet topic models and word embeddings to make lda2vec. *CoRR*, 2016.

[67] Ramesh M. Nallapati, Susan Ditmore, John D. Lafferty, and Kin Ung. Multiscale topic tomography. In *International Conference on Knowledge Discovery & Data Mining (KDD)*, page 520–529, 2007.

[68] David Newman, Youn Noh, Edmund Talley, Sarvnaz Karimi, and Timothy Baldwin. Evaluating topic models for digital libraries. In *Joint Conference on Digital Libraries*, pages 215–224, 2010.

[69] David J Newman and Sharon Block. Probabilistic topic decomposition of an eighteenth-century american newspaper. *Journal of the American Society for Information Science and Technology*, 57(6):753–767, 2006.

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

[71] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39 (2-3):103–134, 2000.

[72] Dan Noyes. The top 20 valuable facebook statistics - updated may 2017. https://zephoria.com/top-15-valuable-facebook-statistics/, 2019. Accessed: 2017-05-05.

[73] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NIPS)*, pages 8024–8035. 2019.

[74] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[75] Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 569–577, 2008.

[76] Pushshift.io. Pushshift.io api documentation. https://pushshift.io/api-parameters/, 2021. Accessed: 2021-03-07.

[77] Jipeng Qiang, Ping Chen, Tong Wang, and Xindong Wu. Topic modeling over short texts by incorporating word embeddings. *CoRR*, 2016.

[78] Jipeng Qiang, Qian Zhenyu, Yun Li, Yunhao Yuan, and Xindong Wu. Short text topic modeling techniques, applications, and performance: A survey. *arXiv preprint arXiv:1904.07695*, 2019.

[79] Xiaojun Quan, Chunyu Kit, Yong Ge, and Sinno Jialin Pan. Short and sparse text topic modeling via self-aggregation. In *International Joint Conference on Artificial Intelligence*, 2015.

[80] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, 2010.

[81] Michael Röder, Andreas Both, and Alexander Hinneburg. Exploring the space of topic coherence measures. In *International Conference on Web Search and Data Mining (WSDM)*, pages 399–408, 2015.

[82] Marco Rossetti, Fabio Stella, and Markus Zanker. Towards explaining latent factors with topic models in collaborative recommender systems. In *International Workshop on Database and Expert Systems Applications*, pages 162–167, 2013.

[83] R Ryan, P. E. Davis-Kean, L. Bode, J Kruger, Z. Mneimneh, and L. Singh. The new dr. spock: Analyzing information provided by parenting-focused twitter accounts. In *[Unpublished paper under review]*, 2020.

[84] Kentaro Sasaki, Tomohiro Yoshikawa, and Takeshi Furuhashi. Online topic model for twitter considering dynamics of user interests and topic trends. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1977–1985, 2014.

[85] Alexandra Schofield, Måns Magnusson, and David Mimno. Pulling out the stops: Rethinking stopword removal for topic models. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, volume 2, pages 432–436, 2017.

[86] Farial Shahnaz, Michael W. Berry, V.Paul Pauca, and Robert J. Plemmons. Document clustering using nonnegative matrix factorization. *Information Processing Management*, pages 373–386, 3 2006.

[87] Lisa Singh, Laila Wahedi, Yanchen Wang, Yifang Wei, Christo Kirov, Susan Martin, Katharine Donato, Yaguang Liu, and Kornraphop Kawintiranon. Blending noisy social media signals with traditional movement variables to predict forced migration. In *International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 1975–1983, 2019.

[88] Lisa Singh, Jonathan Ladd, Josh Pasek, Michael Traugott, Ceren Budak, Stuart Soroka, Jennifer Agiesta, and Grace Sparks. The breakthrough [polling project], 2020.

[89] Jennifer Sleeman, Milton Halem, Tim Finin, Mark Cane, et al. Modeling the evolution of climate change assessment research using dynamic topic models and cross-domain divergence maps. In *AAAI Spring Symposium on AI for Social Good*, 2016.

[90] David Sontag and Daniel M Roy. Complexity of inference in topic models. In *Advances in Neural Information Processing: Workshop on Applications for Topic Models: Text and Beyond*, 2009.

[91] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[92] Stephen A Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377, 2010.

[93] Felipe Viegas, Sérgio Canuto, Christian Gomes, Washington Luiz, Thierson Rosa, Sabir Ribas, Leonardo Rocha, and Marcos André Gonçalves. Cluwords: exploiting semantic word clustering representation for enhanced topic modeling. In *International Conference on Web Search and Data Mining (WSDM)*, pages 753–761, 2019.

[94] Felipe Viegas, Washington Cunha, Christian Gomes, Antônio Pereira, Leonardo Rocha, and Marcos Goncalves. Cluhtm-semantic hierarchical topic modeling based on cluwords. In *Association for Computational Linguistics (ACL)*, pages 8138–8150, 2020.

[95] Daniele Vitale, Paolo Ferragina, and Ugo Scaiella. Classification of short texts by deploying topical annotations. In *European Conference on Information Retrieval (ECIR)*, pages 376–387. Springer, 2012.

[96] Hanna M Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. Evaluation methods for topic models. In *International Conference on Machine Learning (ICML)*, pages 1105–1112, 2009.

[97] Chong Wang, David M. Blei, and David Heckerman. Continuous time dynamic topic models. In *UAI*, 2008.

[98] Chong Wang, John Paisley, and David Blei. Online variational inference for the hierarchical dirichlet process. In *International Conference on Artificial Intelligence and Statistics*, pages 752–760, 2011.

[99] Rui Wang, Deyu Zhou, and Yulan He. Atm: Adversarial-neural topic model. *Information Processing & Management*, 56(6):102098, 2019.

[100] Rui Wang, Xuemeng Hu, Deyu Zhou, Yulan He, Yuxuan Xiong, Chenchen Ye, and Haiyang Xu. Neural topic modeling with bidirectional adversarial training. *arXiv preprint arXiv:2004.12331*, 2020.

[101] Xuerui Wang and Andrew McCallum. Topics over time: A non-markov continuous-time model of topical trends. In *International Conference on Knowledge Discovery & Data Mining (KDD)*, 2006.

[102] Yang Wang and Greg Mori. Human action recognition by semilatent topic models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1762–1774, 2009.

[103] Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. A biterm topic model for short texts. In *The Web Conference (WWW)*, pages 1445–1456, 2013.

[104] Xiaohui Yan, Jiafeng Guo, Shenghua Liu, Xueqi Cheng, and Yanfeng Wang. Learning topics in short texts by non-negative matrix factorization on term correlation matrix. In *SIAM International Conference on Data Mining (SDM)*, 2013.

[105] Tze-I Yang, Andrew Torget, and Rada Mihalcea. Topic modeling on historical newspapers. In *ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 96–104, 2011.

[106] Limin Yao, David Mimno, and Andrew McCallum. Efficient methods for topic model inference on streaming document collections. In *KDD*, 2009.

[107] Xing Yi and James Allan. Evaluating topic models for information retrieval. In *International Conference on Information and Knowledge Management (CIKM)*, pages 1431–1432, 2008.

[108] Xing Yi and James Allan. A comparative study of utilizing topic models for information retrieval. In *European Conference on Information Retrieval (ECIR)*, pages 29–41. Springer, 2009.

[109] Jianhua Yin and Jianyong Wang. A dirichlet multinomial mixture model-based approach for short text clustering. In *International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 233–242. ACM, 2014.

[110] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. Comparing twitter and traditional media using topic models. In *European Conference on Information Retrieval (ECIR)*. Springer, 2011.

[111] Liansheng Zhuang, Haoyuan Gao, Jiebo Luo, and Zhouchen Lin. Regularized semi-supervised latent dirichlet allocation for visual concept learning. *Neurocomputing*, 119: 26–32, 2013.

[112] Yuan Zuo, Junjie Wu, Hui Zhang, Hao Lin, Fei Wang, Ke Xu, and Hui Xiong. Topic modeling of short texts: A pseudo-document view. In *International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 2105–2114, 2016.