2018-12-20

Fine-Grained Topic Models Using Anchor Words

Jeffrey A. Lund
Brigham Young University

Fine-Grained Topic Models Using Anchor Words

Jeffrey A. Lund

A dissertation submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Kevin Seppi, Chair
David Wingate
William Barrett
Michael Jones
Dennis Ng

Department of Computer Science

Brigham Young University

ABSTRACT

Fine-Grained Topic Models Using Anchor Words

Jeffrey A. Lund
Department of Computer Science, BYU
Doctor of Philosophy

Topic modeling is an effective tool for analyzing the thematic content of large collections of text. However, traditional probabilistic topic modeling is limited to a small number of topics (typically no more than hundreds). We introduce fine-grained topic models, which have large numbers of nuanced and specific topics. We demonstrate that fine-grained topic models enable use cases not currently possible with current topic modeling techniques, including an automatic cross-referencing task in which short passages of text are linked to other topically related passages. We do so by leveraging anchor methods, a recent class of topic model based on non-negative matrix factorization in which each topic is anchored by a single word. We explore extensions of the anchor algorithm, including tandem anchors, which relaxes the restriction that anchors be formed of single words. By doing so, we are able to produce anchor-based topic models with thousands of fine-grained topics. We also develop metrics for evaluating token level topic assignments and use those metrics to improve the accuracy of fine-grained topic models.

Keywords: topic modeling, anchor words, cross-reference generation

ACKNOWLEDGMENTS

I understand that the acknowledgments section is likely to become the most widely read section of this entire dissertation. I could of course drone on about how wise and supportive my advisor was (he was), how helpful my committee was (they were), how incredible my collaborators are (they are), how my parents placed me on this path (they did), how my friends have kept me sane through crisis (they have), how my past and future coworker gave me hope for life after school (he did), how I am indebted to the department secretaries for saving my bacon (I am), how my kids think that I make paper for a living (they do), and most of all how patient, long-suffering, and loving my wife is (she really is), but I don't think I could ever do any of that justice. Consequently, what I'll leave here is this:

so long and thanks for all the fish.

Table of Contents

List of Figures ix

List of Tables xiv

List of Listings xvii

Introduction 1

1 Introduction and Overview 2

1.1 Motivation 2

1.2 Existing Topic Models 5

1.2.1 Probabilistic Topic Modeling 6

1.2.2 Anchor Words 12

1.3 Thesis Statement 15

1.4 Overview of Dissertation 15

I Anchor Selection 18

2 Tandem Anchoring: A Multiword Anchor Approach for Interactive Topic Modeling 19

2.1 Introduction 19

2.2 Vanilla Anchor Algorithm 20

2.3 Tandem Anchor Extension 22

2.3.1 Anchor Facets 23
2.3.2 Combining Facets into Pseudowords 24
2.3.3 Finding Topics 26
2.4 High Water Mark for Tandem Anchors 27
2.4.1 Experimental Setup 27
2.4.2 Experimental Results 28
2.4.3 Runtime Considerations 31
2.5 Interactive Anchor Words 32
2.5.1 Interface and User Study 32
2.5.2 Quantitative Results 33
2.5.3 Qualitative Results 35
2.6 Conclusion 37

3 Labeled Anchors: A Scalable, Transparent, and Interactive Classifier 38

3.1 Introduction 38
3.2 Labeled Anchors 40
3.2.1 Vanilla Anchor Words 40
3.2.2 Vector-Space Representations 41
3.2.3 Free Classifier 43
3.2.4 User Interaction 44
3.3 Experimental Results 45
3.4 Conclusion 48

II Token Assignment 49

4 Automatic Evaluation of Local Topic Quality 50

4.1 Introduction 51
4.2 Global Evaluation 53

4.3 Proposed Metrics 55
4.4 Human Evaluations 58
4.4.1 Datasets and Models 58
4.4.2 Crowdsourcing Task 59
4.4.3 Agreement Results 61
4.5 Automated Evaluations 63
4.6 Discussion 64
4.7 Conclusion 65

5 Token Level Topic Assignments **66**

5.1 Introduction 66
5.2 Inference Techniques 68
5.3 Experimental Setup 70
5.4 Results 72
5.5 Conclusion 76

III Application **77**

6 Cross-Referencing Using Fine-Grained Topic Modeling **78**

6.1 Introduction 78
6.2 Methodology 80
6.2.1 Cross-Reference Datasets 80
6.2.2 Baselines for Automated Cross-Reference Generation 82
6.2.3 Topic-based Cross-Referencing 83
6.2.4 Model Selection 84
6.3 Results 86
6.3.1 Metric Comparisons 86
6.3.2 Topic Model Comparison 88

6.3.3 Cost Analysis 90
6.3.4 Qualitative Example 92
6.4 Discussion 94
6.5 Conclusion 94

IV Parallelization 96

7 Parallelization of Anchor Algorithm 97
7.1 Introduction 97
7.2 Parallelization with Mrs 98
7.2.1 Q-construction 98
7.2.2 Topic Recovery 101
7.3 Conclusion 103

8 Mrs: High Performance MapReduce for Iterative and Asynchronous Algorithms in Python 105
8.1 Introduction 106
8.2 Related Work 107
8.3 Synchronous MapReduce 109
8.3.1 Infrequent Checkpointing to Distributed Filesystems 109
8.3.2 Reduce-map Operation 111
8.3.3 Iterative Programming Model 114
8.4 Asynchronous MapReduce Programming Model 118
8.5 Experimental Results 123
8.5.1 Synchronous MapReduce 123
8.5.2 Asynchronous MapReduce 128
8.6 Conclusion 130

Conclusions **132**

9 Conclusions **133**

9.1 Contributions 133

9.2 Future Work 134

A Relationship Between Topic Models and Word Embedding Models **136**

B Examples of Biblical Cross-References **141**

References **153**

List of Figures

2.1 Example of geometric intuition behind tandem anchoring. Words are embedded in V-dimensional cooccurrence space. Notice that neither the word "camera" or "bag" occupy the space which has the occurrence pattern describing the concept of camera bags (i.e., close to "backpack"). However, if we average the two vectors, we end up in the neighborhood which does capture this concept. 23

2.2 Using metadata can improve anchor-based topic models. For all metrics, the unsupervised Gram-Schmidt anchors do worse than creating anchors based on Newsgroup titles (for all metrics except VI, higher is better). For coherence, Gram-Schmidt does better than two functions for combining anchor words, but not the element-wise min or harmonic mean. 29

2.3 Interface for user study with multiword anchors applied to interactive topic modeling. 32

2.4 Classification accuracy and coherence using topic features gleaned from user provided multiword and single word anchors. Grahm-Schmidt anchors are provided as a baseline. For all metrics except VI, higher is better. Except for coherence, multiword anchors are best. 34

2.5 Topic significance for both single word and multiword anchors. In all cases higher is better. Multiword anchors produce topics which are more significant than single word anchors. 34

3.1 Labeled Anchors treats labels as observed words in each labeled documents and updates $\bar{Q}$ under this assumption, creating the additional rows and columns highlighted here. 42
3.2 User study accuracy results comparing accuracy on the development set to the accuracy on the test set. The black horizontal line indicates the baseline accuracy from Supervised Anchors. The black star indicates the initial accuracy using Gram-Schmidt anchors with Labeled Anchors. The blue dots indicate various intermediate steps while editing the anchors. The red pluses are the final states after each user completes the task. 47

4.1 Topic assignments from LDA on a sentence from a Wikipedia document. Notice that even noun-phrases are split in a way which is bewildering to users. 52
4.2 Example of the topic-word matching task. Users are asked to select the topic which best explains the underlined token. 60
4.3 Plot showing human agreement with each model type. CopulaLDA performs slightly worse than LDA. Humans preferred topic assignments from Anchor Words by a wide margin. 62

6.1 Voting interface for cross-reference data from `openbible.info`. 81
6.2 Cross-reference ROC curves for different metrics for cross-reference selection with topics from three different topic models and TSKE as the cross-reference ground-truth. 86
6.3 Cross-reference PRC plots with different metrics for cross-reference selection with topics from three different topic models and TSKE as the cross-reference ground-truth. 87
6.4 Cross-reference ROC curves with different models for cross-reference selection with three different datasets. 89

6.5 Cross-reference PRC plots with different models for cross-reference selection with three different datasets. 89
6.6 Cross-reference cost curves with different models for cross-reference selection with three different datasets. Note the log-log scale. The x-axis denotes the number of cross-references produced by our method, while the y-axis indicates how many of those cross-references are valid according to the human-provided ground truth. 91

7.1 Speedup ratio and parallel efficiency for construction of cooccurrence matrix using Mrs MapReduce using one million Amazon reviews as the text. 100
7.2 Speedup ratio and parallel efficiency for fine-grained topic recovery using Mrs MapReduce. We use 3,000 topics with Gram-Schmidt anchors on the English Standard Version of the Bible. See Chapter 6 for more details on the dataset and topics. 103

8.1 Task dependencies of a typical iterative MapReduce program with (a) standard map (M) and reduce (R) operations, contrasted with (b) combined reduce-map (RM) operations. 112
8.2 Actual task execution traces generated from a sample application (particle swarm optimization, see Section 8.5.1 for details) without and with combined reduce-map tasks. The run with reduce-map operations avoids the overhead of an independent reduce task and completes sooner. The horizontal axis is measured in seconds, with the left and right sides of each box aligning with the task's start and stop times. The number in each box is the key of the map task. 113
8.3 Task dependencies for reduce-map tasks in synchronous and asynchronous iterative MapReduce. Asynchronous MapReduce makes much more efficient use of processors. 119

8.4 Actual task execution traces for PSO with synchronous and Asynchronous MapReduce. The horizontal axis is measured in seconds. 122

8.5 Parallel Efficiency (per iteration) of PSO in MapReduce with a sequence of cumulative optimizations. The x-axis represents the problem size (the number of subiterations in each map task). "Redundant storage" represents the baseline performance, with all data stored to a redundant filesystem and with convergence checks occurring after each iteration. "No redundant storage" shows performance for iterations with data communicated directly between processors. "Concurrent checks" shows further improvements when the convergence check is performed alongside the following iteration's work. "Rare checks" avoids unnecessarily frequent convergence checks. Finally, "reduce-map tasks" agglomerates each pair of reduce and map tasks into a single reduce-map task. 126

8.6 The average throughput (in tasks per second) for synchronous and asynchronous PSO. The number of subiterations per map task vary, with an average of 50 and a standard deviation ranging from 0 to 20. Throughput of the asynchronous implementation is unaffected by task variance and is better even when there is no variance. 129

8.7 The average throughput (in tasks per second) for synchronous and asynchronous PSO with respect to the number of processors. The number of subswarms is equal to the number of processors, and the number of subiterations is 1000. 130

A.1 Some discourse atoms and their nearest 9 words. Each atom is labeled with an arbitrary id number, and the word embeddings are trained on 3 billion tokens of Wikipedia data. Figure taken from Arora et al. [7]. 138

A.2 Some topics represented by the 6 most probable words in a topic. We label each topic using the newsgroup the topic is most heavily associated with. The topics are learned using the anchor-word algorithm on the Twenty Newsgroups dataset. 139

List of Tables

1.1 Examples of topic modeling output on Amazon product reviews, with topics represented by the 10 most probable words in the topic-word distribution. Coarse-grained topics deal with general product types, while fine-grained topics deal with both product types and specific aspects of particular product types. 4

2.1 Three separate attempts to construct a topic concerning camera bags in Amazon product reviews with single word anchors. This example is drawn from preliminary experiments with an author as the user. The term "backpack" is a good anchor because it uniquely identifies the topic. However, both "camera" and "bag" are poor anchors for this topic. 22

2.2 Comparison of topics generated for 20NEWS using various types of anchor words. Users are able to combine words to create more specific topics with tandem anchors. 36

3.1 Runtime for Labeled Anchors and Supervised Anchors on various subsets of Amazon product reviews. Labeled Anchors is dramatically faster than Supervised Anchors and scales to much larger datasets. 45

4.1 Statistics on datasets used in user study and metric evaluation. 58

4.2 Coefficient of determination (r^2) between automated metrics and crowdsourced topic-word matching annotations. We include metrics measuring both local topic quality such as switchp and measures of global topic quality such as coherence and significance. 63

5.1 Statistics on datasets and number of topics used in the study of token level topic assignments. 71
5.2 Topic consistency results. Higher is better. Bold indicates the best result across each dataset and topic set. Coarse topics results is more consistent models. However, regardless of topic granularity, ICM+Word Init yields the most locally consistent topic assignments except for New York Times with fine-grained topics. 72
5.3 Topic significance as measured by divergence from the background document-topic distribution. Higher significance values are better. Bold indicates the best result across dataset and topic sets. Iterated Conditional Modes (ICM) yields the most significant topics, except for New York Times with fine-grained topics. 73
5.4 Classification accuracy results. Bold indicates the best result across dataset. When paired with a good assignment strategy, fine-grained topics improves classification accuracy over standard coarse-grained topics. 74
5.5 Example of topics inferred from the 20 Newsgroups dataset. These particular topics are topics which deal with religious issues. For coarse-grained topics, all three religious topics are shown. For fine-grained, only a small sample of the religious topics are shown. Fine-grained topics are much more nuanced and detailed. 75

6.1 Topics which were attributed to tokens in Isaiah 25:8. See Figure 6.1 for the text of this verse and the most relevant cross-references for this verse. As per usual, each topic is represented as a set of related words, however rather than showing the top n most probable words in each topic, we show the words which had probability greater than $1e-4$ in the topic-word distribution, leading to unequal numbers of words shown for each topic. 92

6.2 Two related passages identified by our cross-referencing system from the works of Jane Austen. Passage 1 is taken from the eighth chapter of *Pride and Prejudice*, and Passage 2 is taken from the fourth chapter of *Emma*. These two passages are a valid cross-reference because they both discuss social standing and family connections in the context of marriage. The connection was found even with their lack of shared words. 93

8.1 Parallel efficiency per iteration of EM for various feature set sizes. As expected, higher feature set sizes lead to lower parallel efficiency, but removing redundant storage significantly helps. Further gains are realized by reducing convergence checks and using the reduce-map operation. 128

List of Listings

7.1 MapReduce program for computing cooccurrence matrices in parallel. 99

7.2 MapReduce program for parallel topic recovery. 102

8.1 The structure of a generic iterative program using a standard iterative-unaware MapReduce API. 116

8.2 The structure of a generic iterative program using a generator-callback MapReduce API for performance and flexibility. 117

8.3 PSO program using a generator-callback MapReduce API. 125

Introduction

We introduce the concept of fine-grained topic modeling, describing several problems which could be solved with such an analysis. We also give a brief overview of existing topic modeling literature, and explain why current techniques are inadaquate for use cases which require fine-grained topical analysis.

Chapter 1

Introduction and Overview

1.1 Motivation

The rate at which we produce new digital text is rapidly increasing. The internet has enabled the layperson to easily distribute text in the form of emails, blog posts, social media posts, and websites. Even traditional media such as books and newspapers are increasingly being published in digital form. Given this incredible volume of text, it is impossible for humans to analyze even a fraction of it without the aid of computers, and there now exists a huge variety of machine learning algorithms which aid in natural language processing and facilitate deeper understanding of large collections of text.

In this dissertation, we focus on problems which can be solved with a nuanced understanding of the topical content of individual documents. For example, with a fine-grained topical understanding of a collection of text, we could find small sets of topically related documents and sentences, or we could predict document metadata using fine-grained topical information. These types problems go beyond information retrieval or simple word concordances in which we look for similar words across the data. Instead we seek to give users a deep topical understanding of individual documents, regardless of whether or not the exact same words are used.

Topic modeling is a well known technique for analyzing documents by their topical content [13]. Topic model output typically includes both topics in the form of topic-word distributions, as well as assignments attributing individual word to a particular topic. Topic models have found a wide variety of applications including document classification [15, 96], document clustering [111, 125], sentiment analysis [106], information retrieval [115], author identification [95], and more. Topic models are often used to facilitate exploratory analysis by presenting users with various visualizations of the resulting topics [26, 39]. Additionally, topic modeling can be viewed as a dimensionality reduction, with the per-token topic assignments serving as features for downstream tasks such as text classification [16, 96].

However, as we will discuss further in Section 1.2, traditional probabilistic topic models result models with relatively few topics. Even when the number of topics is increased, the topic distribution is not uniform across the data but is instead skewed towards a small set of topics with the excess topics remaining unused by the model [113]. While these coarse topics work well for giving a glimpse of the high-level topical content of a dataset as a whole, it is less effective when we want to topically examine individual documents or sentences. For example, if we used document-topic distributions learned using Latent Dirichlet Allocation [16] to find a handful of documents which are most topically related to a single target document in even a moderately sized dataset, we would still need to manually sift through many documents, since each topic must explain hundreds or even thousands of documents. For deeper understanding of individual documents, we require topic modeling algorithms which allow for large numbers of fine-grained topics.

As a solution to problems which require a more nuanced topical understanding of the text, we introduce the idea of fine-grained topic modeling. For such an analysis, we require two things: first we need a topic model with large numbers of topics which are nuanced and specific to small sets of documents or sentences in a dataset. Second, we need a way to correctly assign individual tokens to the various topics, or else the larger numbers of nuanced topics will become less useful when applied to individual documents or sentences.

As an illustration of the difference between traditional coarse-grained topic model output and fine-grained topic model output, consider Table 1.1, which shows topics trained on roughly 40,000 Amazon product reviews. The coarse-grained topics are from a model trained with 40 topics using Latent Dirichlet Allocation, while the fine-grained topics are from a

Coarse-grained Topics
monitor baby hear work static recommend room house clear channel
cable hdmi work price buy quality expensive tv monster cheap
gps garmin turn name street easy unit map direction feature
battery charger charge recharge work time life 4 good long
camera bag lenses backpack canon fit price carry lot lens
mp3 player music easy song good sound sansa battery media
Fine-grained Topics
live city york interfere find nyc channel chicago area pretty
headphone sound noise comfort great cancel hear ear product price
time spend waste bought lot fine money bit worth stop
camera remote sensor front ir shutter pouch infrared delay accept
terrific hesitate easy great excellent good surprise pleased recommend awesome
gift wife daughterinlaw gave enough sooner grandmother present mom idea

Table 1.1: Examples of topic modeling output on Amazon product reviews, with topics represented by the 10 most probable words in the topic-word distribution. Coarse-grained topics deal with general product types, while fine-grained topics deal with both product types and specific aspects of particular product types.

model trained with 4,000 topics using the techniques described in this dissertation. The six representative topics from each model are randomly chosen. The coarse-grained topics tend to deal with general product types. While some fine-grained topics are also concerned with products types such as headphones, others are about a specific aspect of certain products. For example, there is a topic which deals with whether a product is a gift for a female family member, and a topic which describes the remotes included with certain cameras.

The goal of this dissertation is to demonstrate that fine-grained topic modeling not only is possible, but that it enables topic-based use cases which are not currently possible using only traditional coarse-grained topic models. For example, while topic models have been used for document-level classification [96], topic models have not been successfully employed for granular classification in which individual paragraphs or sentences are labeled instead of entire documents.

Another potential application is the problem of aspect level sentiment classification [98]. Rather than attempting to classify the overall sentiment of an entire document, aspect level classification attempts to classify aspects of a document with respect to a specific context. As an example, consider the sentence "overall the food was delicious and we loved the experience, although it was a bit pricey." A general sentiment classifier would likely predict positive sentiment for this sentence, while an aspect level sentiment classification system might identify positive sentiment about food, but a negative sentiment about price. With fine-grained topic modeling, we could accurately determine which aspect is being discussed in specific sentences or phrases, and then make predictions about the various aspects of a document.

While there are many potential applications of fine-grained topic modeling, in this dissertation, we limit our scope by focusing specifically on the problem of automatic cross-reference generation (i.e., finding small sets of the most topically related documents in a large corpus). Historically, the process of generating cross-references resources has been prohibitively expensive. Consequently, large cross-reference resources only exist for the most well studied texts (e.g., religious texts). We use fine-grained topic modeling to propose cross-references which can be reviewed by experts in order to quickly and cheaply produce cross-reference resources for new texts. We demonstrate that this system greatly benefits from the addition of fine-grained topic modeling compared to traditional coarse-grained topic models. We discuss this application of fine-grained topic modeling in Chapter 6.

1.2 Existing Topic Models

Before we discuss our specific contributions, we frame our work by reviewing existing topic modeling literature. We first give an overview of traditional probabilistic topic modeling literature. We then give a brief description of more recent work using anchor words.

We purposefully omit literature dealing with word embeddings. While there are similarities between word embeddings and topic models, the models are trained differently, and those differences make the models useful for different tasks. We discuss the relationship between word embeddings and topic modeling in Appendix A.

1.2.1 Probabilistic Topic Modeling

One of the earliest topic models was Latent Semantic Indexing or LSI. It was first described by Deerwester et al. [31] to aid with information retrieval tasks. Later, the algorithm was formally justified as a topic model [88]. LSI represents documents as a matrix, with each row representing a term-count vector. The algorithm views topic modeling as a matrix factorization problem, relying on singular value decomposition to project documents represented as vectors into a lower dimension topic space. This dimensionality reduction not only improved the empirical performance of information retrieval systems, but was computationally less expensive at query time than existing information retrieval systems at the time.

A probabilistic variant of LSI, appropriately named Probabilistic Latent Semantic Indexing or PLSI, was then developed [49]. Rather than relying on linear algebra, PLSI is a generative model which views topics as probability distributions over words and documents as mixtures of topics. The parameters of PLSI are typically learned using expectation maximization.

Perhaps the most well known work in topic modeling is Latent Dirichlet Allocation or LDA, first proposed by Blei et al. [16]. Like PLSI, LDA views topics as categorical distributions over words, but adds Dirichlet priors to PLSI in order to make the model fully Bayesian. Since LDA is a generative model, we can apply an existing LDA model to new documents without refitting the model to the new data. The Dirichlet priors also allow LDA to smooth topics when data is sparse.

Inference Given a set of observed text documents, the goal with LDA and similar models is typically to compute a *maximum a posteriori* estimate. The idea behind maximizing the posterior probability of the latent topic variables given the observed data is that this will be the setting of latent topic variables which best explains the observed data. In general, computing exact *maximum a posteriori* estimates in models like LDA is NP-hard [103], so practitioners rely on various approximations of the posterior distribution.

Blei et al. [16] originally used variational Bayes to approximate the posterior distribution of LDA. This technique uses the mean field assumption to approximate the true but intractable distribution with an approximation which is trivial to compute. Iteratively minimizing the Kullback-Leibler divergence between the true and the approximate distributions makes the approximate distribution as close as possible to the true posterior. This technique is reasonably fast on moderately sized corpora and tends to yield good *maximum a posteriori* estimates for LDA.

Another popular technique for approximate inference of LDA is a collapsed Gibbs sampler [42]. This technique draws samples from the posterior distribution by sampling values for each individual latent topic variable in the model. The other parameters of the model are marginalized or collapsed in order to reduce the sample complexity of the model using the complete conditional distribution for each individual variable. These updates are performed iteratively to form a Markov chain. Although the chain only updates using the complete conditional, taken together these samples provably come from the true posterior distribution. However, while most Bayesian analysis seeks to characterize the entire posterior distribution, in order to find the topic assignments that best explain the data, we desire a *maximum a posteriori* estimate. To do so, typically the sampler is run until it has stabilized, and a representative sample is selected using the intuition that the sampler tends towards regions of high probability and is likely to be close to a mode in the posterior distribution.

Other techniques such as expectation propagation have also been explored [77]. With respect to held-out perplexity, which is defined as $2^{H(p)}$, where $H(p)$ is the entropy of the distribution over held-out data, each of these inference algorithms tends to perform roughly the same when coupled with hyperparameter optimization [8]. Typically, hyperparameter optimization is performed using fixed-point techniques [112]. Unfortunately, while hyperparameter optimization is needed to improve the quality of the *maximum a posteriori* estimates, it also tends to reduce the number of meaningful topics in the model [8]. For example, Wallach et al. [113] found that increasing the number of topics beyond 25 had little effect on datasets such as 20 newsgroups, despite the number of documents. Consequently, the usefulness of LDA and similar models is diminished when applied to problems requiring fine-grained topic modeling, as each topic, or rather, each of the few topics which are actually utilized by the model, must explain many documents.

When speed is required, various other inference algorithms have been proposed. For example, SparseLDA modifies the existing Gibbs sampler by binning low probability topic assignments so they can be considered in aggregate [118]. In the presence of parallel computational resources, a parallel Gibbs sampler has been developed [102]. While this inference scheme technically does not produce correct Markov chains while sampling, it has been shown to achieve results comparable to a more correct but serial Gibbs sampler. Online learning for LDA has also been developed [47].

Evaluation Since topics are categorical distributions over the entire corpus vocabulary, understanding model output can be somewhat daunting for users. Thus visualization is useful to help users explore topical trends in data. An early effort in topic visualization was the Topic Browser [39]. Other popular topic visualization software include Termite [26] and a system by Chaney and Blei [22]. Each of these systems allows users to visualize topic as sets of related words, as well as browse documents which correspond to particular topics. When paired with appropriate document data, other topical trends can be visualized. For example, Topic Browser allows users to plot the prevalence of topics over time.

Barring topic visualization systems, a common way to represent topics to users is by presenting topics as a set of the n most probable words in the topic-word distribution. Consequently, a popular way of evaluating topic models is with topic coherence, which is defined as:

$$\sum_{v_1, v_2 \in V} log \frac{D(v_1, v_2) + \epsilon}{D(v_2)}, \tag{1.1}$$

where the function $D(v_1, v_2)$ is the co-document frequency of word types v_1 and v_2, and $D(v_2)$ is the document frequency of word type v_2 [76]. Alternatively, coherence can be defined with respect to the pairwise word cooccurrences in an external dataset such as Wikipedia [83].

Topic coherence has been shown to correlate with human evaluations of topic quality. This is an advantage compared to earlier automated evaluations such as held-out perplexity, which have since been shown to actually be negatively correlated with human evaluations of topic coherency [23]. On the other hand, with topic coherence some care must be taken with regards to the choice of how many of the most probable words to use in the coherence calculation [59].

Other work on automatic evaluation of topics includes metrics for evaluating topic significance [1]. Topic significance is evaluated in three ways. First, the topic-word distributions are compared to the uniform distribution over words, with the idea that a significant topic should focus on a few specific terms rather than many terms uniformly. Second, the distance between the topic-word distributions and the vacuous, or background distribution of words in the entire data is measured, with the idea that significant topics should not reflect the entire data, but specific thematic trends in the data. Finally, the document-topic distribution is compared to the uniform distribution over documents, using the intuition that significant topics should be present in a smaller number of documents, rather than being a background topic found throughout the entire corpus of documents.

Despite the usefulness of automated metrics of topic quality, perhaps the most important way to evaluate topic models is by measuring their performance for use in the intended downstream tasks. For example, topic modeling can be viewed as a dimensionality reduction technique, with topics serving as features for classification tasks. Thus it is common to evaluate topic models such as LDA by their performance as classification features [16, 96]. Topic models have also been applied to the field of information retrieval to improve ad-hoc retrieval [115]. Another task topic models have been successful at is word sense disambiguation [18].

One area in which evaluation techniques have been lacking is in token level topic assignments. As evidenced by recent work such as CopulaLDA [10], there is interest in improving the quality of token level topic assignments. However, even this work relies on traditional global topic-word distributions for evaluation. We address this shortcoming and explore ways of evaluating local topic model quality in Chapter 4.

LDA Extensions LDA is also easily extended to create topic models which perform specialized tasks. These models can be highly specific to a particular task or dataset. For example, the Capsule Model was developed for aiding historical research by topically analyzing millions of U.S. State Department cables, taking into account not only the text, but the entities, dates, and events in each cable. For more generic types of supervised learning, sLDA is a supervised extension of LDA which allows topics to be inferred from document text as well as document metadata [15]. Once learned, an sLDA model can be used to both infer topics and predict missing metadata on new data. A similar approach has been used to perform topic-based multi-label classification [96]. The Multi-Aspect Sentiment model [106] and Labeled LDA [91] accomplish a similar result for sentiment and tags respectively. The Author-Topic Model generates topics which utilize authorship information during inference [95].

Other extensions of LDA seek to improve the topical structure in some way. For example the Correlated Topic Model [14] and the Pachinko Allocation [61] change the model structure so that inter-topic correlations are explicitly modeled. The Pachinko Allocation model is useful in particular as it allows practitioners to define an arbitrary directed acyclic graph structure for topics, modeling topic correlations at each level of the graph. In fact, LDA is a special case of Pachinko Allocation. Similarly, Hierarchical LDA [41] changes the model structure by using a nested Chinese restaurant process to model topics in a hierarchical structure.

Another type of modification to LDA is to reexamine the distribution family used to model topics. LDA relies on the Dirichlet-Multinomial conjugate pair to model topic-word distributions, but other distributions could be used instead. For example, the Spherical Admixture Model employs the same basic structure of LDA, but uses a von Mises-Fisher distribution over words on a unit hypersphere [92]. This allows us to measure document similarity using cosine distance, and allows topics to assign negative weight to words. Alternatively, we can replace the traditional Dirichlet-Multinomial by modeling log-frequencies in order to combine multiple facets (including topics) of text into an additive generative model [35].

Integrating syntax into topic models has also proven to be useful. Both the Syntactic Topic Model [17] and HMM-LDA [43] integrate short-range syntactic dependencies with long-range topic dependencies and can be used to jointly infer topics with other syntactic features such as part-of-speech tags. More recent work uses copulas to model short-range topic dependencies in order to improve token-level topic assignments [10].

Interactive Topic Modeling Another important line of research related to topic modeling is the idea of interactive topic modeling. Combining topic visualization with topic inference, interactive topic models add human input into the loop during the inference process. By adding human interactivity, we are able to inject user specific domain knowledge into the model, or custom tailor the model for a user-specified analysis.

The Interactive Topic Model (ITM) is perhaps the most well known example of a topic model which incorporates human guidance during inference [51]. Rather than modeling topics as distributions over words, the ITM models topics as Dirichlet forests of words. The structure and priors of the Dirichlet forests encode user-specified word constraints. For example, two words could be placed into a forest with an appropriate prior to form a "must-link" constraint which requires a topic to highly weight either both words, or neither word.

In order for a model to be interactive, topic inference must be fast. If the time between updates is too great, cognitive load on the user increases and the interaction suffers [28]. For the ITM, a modified Gibbs sampling scheme based on SparseLDA [118] has been proposed [50]. Alternatively, Iterated Conditional Modes [11] can be used to quickly find a locally optimal mode in the posterior in order to quickly update an ITM model [65].

Topic models are also useful for a variety of tasks. A key insight as to why LDA has garnered so much interest and inspired so many derivative topic models is the power of model based learning. Rather than simply feeding data to some existing machine learning algorithm, we create a model which encodes our assumptions about the data into a generative story. That generative story can include not only latent topics, but any number of other observable document metadata. This model is then paired with a generic inference algorithm in order to create a new machine learning algorithm which is custom tailored to solve a specific problem. This makes probabilistic topic modeling extremely flexible and powerful, and relatively easy for practitioners to adapt for specialized tasks.

1.2.2 Anchor Words

While probabilistic topic models are easily adapted to a wide variety of tasks, and efficient approximate inference algorithms have been developed to learn such models, the runtime complexity of inference tends to scale linearly with the size of the data. Consequently, topic modeling literature tends to work with datasets consisting of tens of thousands of documents. However, for web scale sized datasets, inference speed becomes an issue for probabilistic topic models.

Fortunately, a new class of topic models called the anchor algorithm has emerged which promises much more scalable inference. Like LSI [31], the anchor algorithm views topic modeling as a matrix factorization problem. However, LSI relies on singular value decomposition whereas the anchor algorithm utilizes non-negative matrix factorization. Arora et al. [5] argue that topic modeling based on singular value decomposition must have one of two limitations: either each document must contain only a single topic, or we are only able to recover the span of the topic vectors (rather than the topics themselves). Non-negative matrix factorization overcomes these limitations, and lets us view topics probabilistically if we normalize the resulting topic matrix.

The goal of the anchor algorithm is to compute an unknown $V \times K$ word-topic matrix $\boldsymbol{A}$ with non-negative entries, where V is the vocabulary size, and K is the desired number of topics. Once column-normalized, $\boldsymbol{A}_{vk}$ encodes the conditional probability of observing word type v given topic k. There is also a topic-document matrix $\boldsymbol{W}$ of dimension $K \times D$, where

D is the number of documents in our dataset. The dth column of $\boldsymbol{W}$ gives the proportion of each topic in the dth document. Since the observed $V \times D$ term-document matrix $\boldsymbol{M}$ is equal to $\boldsymbol{AW}$, non-negative matrix factorization can be used to recover the topic matrix $\boldsymbol{A}$.

In general, non-negative matrix factorization is an NP-Hard problem, even when the inner dimension K is small [4]. Consequently, when non-negative matrix factorization is required, typically we rely on approximations which minimize $\| \boldsymbol{M} - \boldsymbol{AW} \|_F$, where $\|\|_F$ is the Frobenius norm [24]. However, Arora et al. [4] also show that under certain conditions of separability, non-negative matrix factorization can be computed exactly in polynomial time.

This separability assumption states that for each topic k, there exists at least one anchor word which has non-zero probability only in that topic: $\boldsymbol{A}_{v,k} >> \boldsymbol{A}_{v,j} \forall j \neq k$. This separability assumption was originally described by Donoho and Stodden [34] for image segmentation, but has been shown to hold for many machine learning applications, including topic modeling [5]. Anchor words make computing the topic matrix $\boldsymbol{A}$ tractable because the occurrence pattern of the anchor words across documents must mirror the occurrence pattern of the topics across documents.

To recover topic matrix $\boldsymbol{A}$ using anchor words, we first compute a $V \times V$ row-normalized word cooccurrence matrix $\boldsymbol{Q}$, where $\boldsymbol{Q}_{i,j}$ is the conditional probability $p(w_j \,|\, w_i)$: seeing word type w_j after having seen type w_i in the same document. Intuitively, $\boldsymbol{Q}$ can be thought of as $\boldsymbol{MM}^T$, although the actual details of its construction are given by Arora et al. [6].

We then select K anchor words $\{g_1 \ldots g_K\}$ using a form of the Gram-Schmidt process described in Arora et al. [6]. Each word corresponds to a row in $\boldsymbol{Q}$. We view each row of $\boldsymbol{Q}$ as a vector in V-dimensional space. The Gram-Schmidt process greedily chooses points which maximize the volume of the simplex formed by the chosen points. Each of these points correspond to a word type, so we choose points to serve as anchor words which can represent more words per document.

Once we have the set of anchor words g, we compute the probability of a topic given a word (the inverse of the conditioning in $\boldsymbol{A}$). This matrix, $\boldsymbol{C}$, is defined row-wise for each word

i as $\boldsymbol{C}_{i\cdot}^* = \underset{C_{i\cdot}}{argmin}\ D_{KL}\left(\boldsymbol{Q}_{i\cdot} \| \sum_{k=1}^{K} \boldsymbol{C}_{ik}\boldsymbol{Q}_{g_k\cdot}\right)$, where D_{KL} is Kullback-Leibler divergence. Thus each non-anchor word is represented as a convex combination of the anchor words. Solving each row of $\boldsymbol{C}$ is fast using exponentiated gradient descent and is embarrassingly parallel. Since $\boldsymbol{C}$ is the inverse conditioning of $\boldsymbol{A}$, we can recover $\boldsymbol{A}$ using Bayes' rule by multiplying $\boldsymbol{C}$ by the word probabilities (the word probabilities are easily obtained by computing $\boldsymbol{Q}\vec{1}$).

The training time using the anchor algorithm can be orders of magnitude faster than methods based on probabilistic modeling. The construction of $\boldsymbol{Q}$ requires only a single pass over the data and can be pre-computed as part of the data import. Once $\boldsymbol{Q}$ is constructed, actual topic inference scales with the size of $\boldsymbol{Q}$, which in turn, is dependent on the size of the vocabulary V. Following Heaps' law, the vocabulary size V tends to grow exponentially slower than the number of documents D [46]. Consequently, at a certain dataset size, topic recovery with the anchor algorithm is essentially unaffected by increased scale.

In contrast, topic inference using traditional model-based approaches such as Latent Dirichlet Allocation [16] typically requires multiple passes over the entire data to infer topics. For example, using a Gibbs sampler to estimate an LDA model requires one to iteratively sample each variable in the model multiple times (often hundreds) until the sampler stabilizes. Techniques such as Online LDA [47] or Stochastic Variation Inference [48] could improve this to a single pass over the entire data. Nevertheless, topic inference for model-based approaches scales with the size of the data.

Since the introduction of the anchor algorithm, some minor improvements to the algorithm have been proposed. Lee and Mimno [60] propose a better anchor selection algorithm based on computing the convex hull of a t-SNE [68] projected version of $\boldsymbol{Q}$. This method helps find more salient words to become anchors, compared to the somewhat eccentric words chosen by the greedy Gram-Schmidt algorithm. Nguyen et al. [85] also add robustness to the anchor algorithm by adding regularization to the objective function when computing the rows of the coefficient matrix $\boldsymbol{C}$.

While the scalability of the anchor algorithm is attractive compared to model based topic inference, it is not a drop-in replacement for probabilistically driven work, nor does it directly accomplish our goal of fine-grained topic modeling with an eye towards topical analysis of individual documents. The existing Gram-Schmidt based technique for finding anchors tends to find eclectic words as anchors, making it unsuited for fine-grained topic modeling. Furthermore, without a way to measure and improve the word level topic assignments, fine-grained topic models are less useful because the analysis of individual documents is less accurate.

1.3 Thesis Statement

We introduce fine-grained topic modeling, which combines large numbers of highly nuanced topics with correct token level topic assignments. Fine-grained topic modeling enables topic-based use cases, such as finding small sets of topically related documents, which are not feasible with current topic modeling techniques.

1.4 Overview of Dissertation

Because traditional probabilistic topic modeling based on LDA cannot support a large number of topics and has problems with token level topic assignment accuracy, we turn to anchor methods to solve the problem of fine-grained topic modeling. This dissertation presents a variety of contributions on this front in three different parts:

Part I: Anchor Selection Our first set of contributions revolves around anchor selection. The anchor algorithm as described by Arora et al. [6] requires that each topic be anchored by a single anchor words which uniquely identifies the topic (meaning it has non-zero probability) in that topic only. Furthermore, if a large number of anchors is used, the anchors tend to become strange and esoteric [60]. We make two major contributions improving anchor selection.

First, in Chapter 2 we relax the restriction that anchor words be a single word. The anchor algorithm works by representing words in a vector-space and then choosing certain words (i.e., points in that space) to anchor each topic. We allow multiple words to form an anchor by averaging the vector-space representation of each word to pick a central point to anchor a topic.

We present this contribution as a way to extend the anchor words algorithm to allow interactive topic modeling. This contribution is useful for fine-grained topic modeling since it lets us select large numbers of nuanced anchors, instead of being restricted to the few points in vector-space which correspond to a single word.

In Chapter 3, we also present a way to allow outside information such as document metadata or other supervised labels to inform anchor selection. This work, which we call labeled anchor words, provides a way to extend anchor words to include a transparent and interactive topic-based document classifier. However, we claim that allowing metadata to influence anchor selection can also be used to enable fine-grained topic modeling by refining anchors to be more nuanced as they better reflect not only the text, but also important metadata attributes.

Part II: Token Assignment While selecting better anchors enables us to learn more nuanced topics, this nuance is lost if the token level topic assignments are inaccurate. We make two contributions towards improving token-level (or local) topic model quality.

First, in Chapter 4, we explore local topic model evaluation. Several models have been proposed which claim to improve the topic assignments of LDA (e.g., CopulaLDA [10], SentenceLDA[9]). However, these models have only been evaluated globally without respect to the actual token level topic assignments.

To rectify this problem, we design a crowdsourcing task which can elicit human evaluation of topic quality. We use this task on a large number of topic models on three different datasets. We then propose a variety of potential automated metrics and compute correlation between the automated metrics and the human evaluations. With an automated metric which correlates well with human evaluations, we can then evaluate new topic models with respect to local topic quality.

We use this new metric in Chapter 5, in which we explore the best way to assign words to topics using topics learned from anchor methods. Since the anchor method only recovers the global topic-word probabilities, typically practitioners use LDA with fixed topics to make the topic assignments [86]. Since LDA is known to have issues with local topic quality [10] and armed with automated metrics to evaluate local topic quality, we investigate various methods of computing the local topic assignments using topics learned by anchor words.

Part III: Application Finally, in Chapter 6 we apply what we have learned to the problem of cross-referencing. This problem is hard because it either requires annotators to evaluate n^2 potential references or be intimately familiar with the text to the point that they can recall related passages of text while reading another passage. As an alternative, we can use topic modeling to find related passages of text. We demonstrate that anchor-based fine-grained topic models are able to significantly reduce the cost of producing cross-reference resources compared to traditional coarse-grained topic modeling.

Part IV: Parallelization As one final contribution, we explore parallelization of parts of the anchor algorithm in Chapter 7. While this is not strictly necessary for the algorithm or our experiments with fine-grained topic modeling, it has proved useful for the completion of this dissertation by enabling us to take advantage of our parallel hardware. Our implementation utilizes Mrs, a Python implementation of MapReduce. We describe some of our modifications to Mrs which make it especially suitable for scientific computing in general in Chapter 8.

Part I

Anchor Selection

With anchor-based topic modeling, each topic is anchored by a word which uniquely identifies the topic. Our first contribution towards fine-grained topic modeling is two published papers which improve the process of anchor selection. In Chapter 2, we relax the constraint that each topic be anchored by a single word and instead allow topics to be anchored by "tandem anchors" which are composed of multiple words. This work is presented as an extension to the anchor words algorithm for interactive topic modeling, but as we'll see later in Chapter 6, the technique can also be used to improve fine-grained topic models. In Chapter 3, we further improve anchor selection by presenting a method for incorporating document metadata such as ratings on product reviews into the anchor selection process. This work is presented as a supervised extension to anchor words, and allows anchor-based topic models to take advantage of metadata during anchor selection and topic inference.

Chapter 2

Tandem Anchoring: A Multiword Anchor Approach for Interactive Topic Modeling

Published in Proceedings of the Association for Computational Linguistics 2016 [66]

Abstract

Interactive topic models are powerful tools for understanding large collections of text. However, existing sampling-based interactive topic modeling approaches scale poorly to large data sets. Anchor methods, which use a single word to uniquely identify a topic, offer the speed needed for interactive work but lack both a mechanism to inject prior knowledge and lack the intuitive semantics needed for user-facing applications. We propose combinations of words as anchors, going beyond existing single word anchor algorithms—an approach we call "Tandem Anchors". We begin with a synthetic investigation of this approach then apply the approach to interactive topic modeling in a user study and compare it to interactive and non-interactive approaches. Tandem anchors are faster and more intuitive than existing interactive approaches.

2.1 Introduction

Topic models distill large collections of text into topics, giving a high-level summary of the thematic structure of the data without manual annotation. In addition to facilitating discovery of topical trends [39], topic modeling is used for a wide variety of problems including document classification [96], information retrieval [115], author identification [95], and sentiment analysis [106]. However, the most compelling use of topic models is to help users understand large datasets [26].

Interactive topic modeling [52] allows non-experts to refine automatically generated topics, making topic models less of a "take it or leave it" proposition. Including human input during training improves the quality of the model and allows users to guide topics in a specific way, custom tailoring the model for a specific downstream task or analysis.

The downside is that interactive topic modeling is slow—algorithms typically scale with the size of the corpus—and require non-intuitive information from the user in the form of must-link and cannot-link constraints [3]. We address these shortcomings of interactive topic modeling by using an interactive version of the *anchor words* algorithm for topic models.

The anchor algorithm [6] is an alternative topic modeling algorithm which scales with the number of unique word types in the data rather than the number of documents or tokens (Section 2.2). This makes the anchor algorithm fast enough for interactive use, even in web-scale document collections.

A drawback of the anchor method is that anchor words—words that have high probability of being in a *single* topic—are not intuitive. We extend the anchor algorithm to use multiple anchor words in tandem (Section 2.3). Tandem anchors not only improve interactive refinement, but also make the underlying anchor-based method more intuitive.

For interactive topic modeling, tandem anchors produce higher quality topics than single word anchors (Section 2.4). Tandem anchors provide a framework for fast interactive topic modeling: users improve and refine an existing model through multiword anchors (Section 2.5). Compared to existing methods such as Interactive Topic Models [52], our method is much faster.

2.2 Vanilla Anchor Algorithm

The anchor algorithm computes the topic matrix $\boldsymbol{A}$, where $\boldsymbol{A}_{v,k}$ is the conditional probability of observing word v given topic k, e.g., the probability of seeing the word "lens" given the camera topic in a corpus of Amazon product reviews. Arora et al. [4] find these probabilities by assuming that every topic contains at least one 'anchor' word which has a non-zero probability only in that topic. Anchor words make computing the topic matrix $\boldsymbol{A}$ tractable because the occurrence pattern of the anchor word mirrors the occurrence pattern of the topic itself.

To recover the topic matrix $\boldsymbol{A}$ using anchor words, we first compute a $V \times V$ cooccurrence matrix $\boldsymbol{Q}$, where $\boldsymbol{Q}_{i,j}$ is the conditional probability $p(w_j \,|\, w_i)$ of seeing word type w_j after having seen w_i in the same document. A form of the Gram-Schmidt process on $\boldsymbol{Q}$ finds anchor words $\{g_1 \dots g_k\}$ [6].

Once we have the set of anchor words, we can compute the probability of a topic given a word (the inverse of the conditioning in $\boldsymbol{A}$). This coefficient matrix $\boldsymbol{C}$ is defined row-wise for each word i

$$\boldsymbol{C}^*_{i,\cdot} = \underset{C_{i,\cdot}}{argmin}\ D_{KL}\left(\boldsymbol{Q}_{i,\cdot} \middle\| \sum_{k=1}^{K} \boldsymbol{C}_{i,k}\boldsymbol{Q}_{g_k,\cdot}\right), \tag{2.1}$$

which gives the best reconstruction (based on Kullback-–Leibler divergence D_{KL}) of non-anchor words given anchor words' conditional probabilities. For example, in our product review data, a word such as "battery" is a convex combination of the anchor words' contexts ($\boldsymbol{Q}_{g_k,\cdot}$) such as "camera", "phone", and "car". Solving each row of C is fast and is embarrassingly parallel. Finally, we apply Bayes' rule to recover the topic matrix $\boldsymbol{A}$ from the coefficient matrix $\boldsymbol{C}$.

The anchor algorithm can be orders of magnitude faster than probabilistic inference [6]. The construction of $\boldsymbol{Q}$ has a runtime of $O(DN^2)$ where D is the number of documents and N is the average number of tokens per document. This computation requires only a single pass over the data and can be pre-computed for interactive use-cases. Once $\boldsymbol{Q}$ is constructed, topic recovery requires $O(KV^2 + K^2VI)$, where K is the number of topics, V is the vocabulary size, and I is the average number of iterations (typically 100-1000). In contrast, traditional topic model inference typically requires multiple passes over the entire data. Techniques such as Online LDA [47] or Stochastic Variation Inference [48] improves this to a single pass over

Anchor	Top Words in Topics
backpack	backpack camera lens bag room carry fit cameras equipment comfortable
camera	camera lens pictures canon digital lenses batteries filter mm photos
bag	bag camera diaper lens bags genie smell room diapers odor

Table 2.1: Three separate attempts to construct a topic concerning camera bags in Amazon product reviews with single word anchors. This example is drawn from preliminary experiments with an author as the user. The term "backpack" is a good anchor because it uniquely identifies the topic. However, both "camera" and "bag" are poor anchors for this topic.

the entire data. However, from Heaps' law [46] it follows that $V^2 \ll DN$ for large datasets, leading to much faster inference times for anchor methods compared to probabilistic topic modeling. Further, even if online inference techniques were to be adapted to incorporate human guidance, a single pass is not tractable for interactive use.

2.3 Tandem Anchor Extension

Single word anchors can be opaque to users. For an example of bewildering anchor words, consider a camera bag topic from a collection of Amazon product reviews (Table 2.1). The anchor word "backpack" may seem strange. However, this dataset contains nothing about regular backpacks; thus, "backpack" is unique to camera bags. Bizarre, low-to-mid frequency words are often anchors because anchor words must be *unique* to a topic; intuitive or high-frequency words cannot be anchors if they have non-zero probability in *any other topic*.

The anchor selection strategy can mitigate this problem to some degree. For example, rather than selecting anchors using an approximate convex hull in high-dimensional space, we can find an exact convex hull in a low-dimensional embedding [60]. This strategy will produce more salient topics but still makes it difficult for users to manually choose unique anchor words for interactive topic modeling.

If we instead ask users to give us representative words for this topic, we would expect combinations of words like "camera" and "bag." However, with single word anchors we must choose a single word to anchor each topic. Unfortunately, because these words might appear

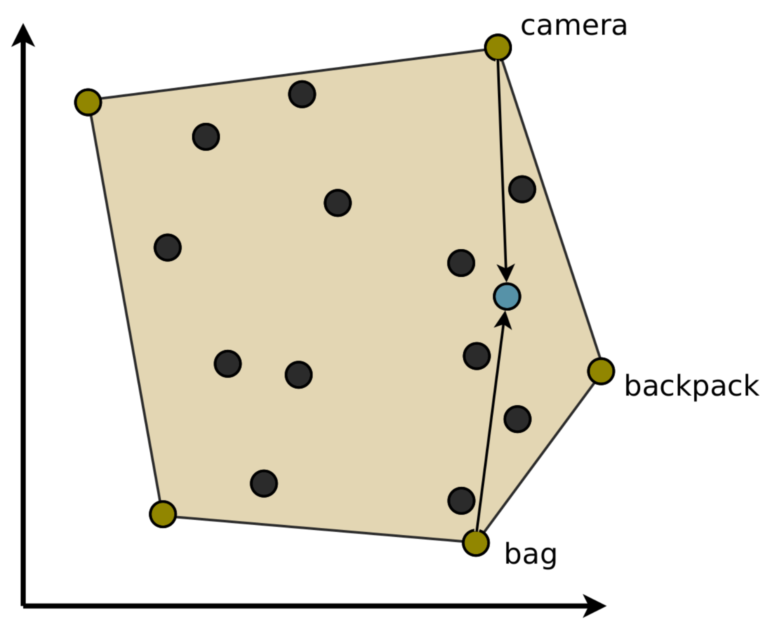


Figure 2.1: Example of geometric intuition behind tandem anchoring. Words are embedded in V-dimensional cooccurrence space. Notice that neither the word "camera" or "bag" occupy the space which has the occurrence pattern describing the concept of camera bags (i.e., close to "backpack"). However, if we average the two vectors, we end up in the neighborhood which does capture this concept.

in multiple topics, individually they are not suitable as anchor words. The anchor word "camera" generates a general camera topic instead of camera bags, and the topic anchored by "bag" includes bags for diaper pails (Table 2.1).

Instead, we need to use sets of representative terms as an interpretable, parsimonious description of a topic. This section discusses strategies to build anchors from multiple words and the implications of using multiword anchors to recover topics. This extension not only makes anchors more interpretable but also enables users to manually construct effective anchors in interactive topic modeling settings.

2.3.1 Anchor Facets

We first need to turn words into an anchor. If we interpret the anchor algorithm geometrically, each row of $\boldsymbol{Q}$ represents a word as a point in V-dimensional space. We then model each point as a convex combination of anchor words to reconstruct the topic matrix $\mathbf{A}$ (Equation 2.1).

Instead of individual anchor words (one anchor word per topic), we use anchor **facets**, or sets of words that describe a topic. The facets for each anchor form a new **pseudoword**, or an invented point in V-dimensional space (described in more detail in Section 2.3.2).

While these new points do not correspond to words in the vocabulary, we can express non-anchor words as convex combinations of pseudowords. To construct these pseudowords from their facets, we combine the co-occurrence profiles of the facets. These pseudowords then augment the original cooccurrence matrix $\boldsymbol{Q}$ with K additional rows corresponding to synthetic pseudowords forming each of K multiword anchors. We refer to this augmented matrix as $\boldsymbol{S}$. The rest of the anchor algorithm proceeds unmodified.

Our augmented matrix $\boldsymbol{S}$ is therefore a $(V + K) \times V$ matrix. As before, V is the number of token types in the data and K is the number of topics. The first V rows of $\boldsymbol{S}$ correspond to the V token types observed in the data, while the additional K rows correspond to the pseudowords constructed from anchor facets. Each entry of $\boldsymbol{S}$ encodes conditional probabilities so that $S_{i,j}$ is equal to $p(w_i \,|\, w_j)$. For the additional K rows, we invent a cooccurrence pattern that can effectively explain the other words' conditional probabilities.

This modification is similar in spirit to supervised anchor words [86]. This supervised extension of the anchor words algorithm adds columns corresponding to conditional probabilities of metadata values after having seen a particular word. By extending the vector-space representation of each word, anchor words corresponding to metadata values can be found. In contrast, our extension does not add dimensions to the representation, but simply places additional points corresponding to pseudoword words in the vector-space representation.

2.3.2 Combining Facets into Pseudowords

We now describe more concretely how to combine anchor facets to describe the cooccurrence pattern of our new pseudoword anchor. In tandem anchors, we create vector representations that combine the information from anchor facets. Our anchor facets are $\mathcal{G}_1 \ldots \mathcal{G}_K$, where $\mathcal{G}_k$ is a set of anchor facets which will form the kth pseudoword anchor. The pseudowords are

$g_1 \ldots g_K$, where g_k is the pseudoword from $\mathcal{G}_k$. These pseudowords form the new rows of $\boldsymbol{S}$. We give several candidates for combining anchors facets into a single multiword anchor; we compare their performance in Section 2.4.

Vector Average An obvious function for computing the central tendency is the vector average. For each anchor facet,

$$\boldsymbol{S}_{g_k,j} = \sum_{i \in \mathcal{G}_k} \frac{\boldsymbol{S}_{i,j}}{|\mathcal{G}_k|}, \tag{2.2}$$

where $|\mathcal{G}_k|$ is the cardinality of $\mathcal{G}_k$. Vector average makes the pseudoword $\boldsymbol{S}_{g_k,j}$ more central, which is intuitive but inconsistent with the interpretation from Arora et al. [6] that anchors should be extreme points whose linear combinations explain more central words.

Or-operator An alternative approach is to consider a cooccurrence with *any* anchor facet in $\mathcal{G}_k$. For word j, we use De Morgan's laws to set

$$\boldsymbol{S}_{g_k,j} = 1 - \prod_{i \in \mathcal{G}_k} (1 - \boldsymbol{S}_{i,j}). \tag{2.3}$$

Unlike the average, which pulls the pseudoword inward, this or-operator pushes the word outward, increasing each of the dimensions. Increasing the volume of the simplex spanned by the anchors explains more words.

Element-wise Min Vector average and or-operator are both sensitive to outliers and cannot account for polysemous anchor facets. Returning to our previous example, both "camera" and "bag" are bad anchors for camera bags because they appear in documents discussing other products. However, if both "camera" and "bag" are anchor facets, we can look at an *intersection* of their contexts: words that appear with both. Using the intersection, the cooccurrence pattern of our anchor facet will only include terms relevant to camera bags.

Mathematically, this is an element-wise min operator,

$$\boldsymbol{S}_{g_k,j} = \min_{i \in \mathcal{G}_k} \boldsymbol{S}_{i,j}. \tag{2.4}$$

This construction, while perhaps not as simple as the previous two, is robust to words which have cooccurrences which are not unique to a single topic.

Harmonic Mean Leveraging the intuition that we should use a combination function which is both centralizing (like vector average) and ignores large outliers (like element-wise min), the final combination function is the element-wise harmonic mean. Thus, for each anchor facet

$$\boldsymbol{S}_{g_k,j} = \sum_{i \in \mathcal{G}_k} \left(\frac{\boldsymbol{S}_{i,j}^{-1}}{|\mathcal{G}_k|} \right)^{-1}. \tag{2.5}$$

Since the harmonic mean tends towards the lowest values in the set, it is not sensitive to large outliers, giving us robustness to polysemous words.

2.3.3 Finding Topics

After constructing the pseudowords of $\boldsymbol{S}$ we then need to find the coefficients $\boldsymbol{C}_{i,k}$ which describe each word in our vocabulary as a convex combination of the multiword anchors. Like standard anchor methods, we solve the following for each token type:

$$\boldsymbol{C}_{i,\cdot}^{*} = \underset{\boldsymbol{C}_{i,\cdot}}{argmin}\ D_{KL} \left(\boldsymbol{S}_{i,\cdot} \middle\| \sum_{k=1}^{K} \boldsymbol{C}_{i,k} \boldsymbol{S}_{g_k,\cdot} \right). \tag{2.6}$$

Finally, we appeal to Bayes' rule to we recover the topic-word matrix $\boldsymbol{A}$ from the coefficients of $\boldsymbol{C}$.

The correctness of the topic recovery algorithm hinges upon the assumption of separability. Separability means that the occurrence pattern across documents of the anchor words across the data mirrors that of the topics themselves. For single word anchors, this has been observed to hold for a wide variety of data [5]. With our tandem anchor extension, we make similar assumptions as the vanilla algorithm, except with pseudowords constructed from anchor facets. So long as the occurrence pattern of our tandem anchors mirrors that of the underlying topics, we can use the same reasoning as Arora et al. [4] to assert that we can provably recover the topic-word matrix $\boldsymbol{A}$ with all of the same theoretical guarantees of complexity and robustness. Furthermore, the runtime analysis given by Arora et al. [6] applies to tandem anchors.

If desired, we can also add further robustness and extensibility to tandem anchors by adding regularization to Equation 2.6. Regularization allows us to add something which is mathematically similar to priors, and has been shown to improve the vanilla anchor word algorithm [85]. We leave the question of the best regularization for tandem anchors as future work, and focus our efforts on solving the problem of interactive topic modeling.

2.4 High Water Mark for Tandem Anchors

Before addressing interactivity, we apply tandem anchors to real world data, but with anchors gleaned from metadata. Our purpose is twofold. First, we determine which combiner from Section 2.3.2 to use in our interactive experiments in Section 2.5 and second, we confirm that well-chosen tandem anchors can improve topics. In addition, we examine the runtime of tandem anchors and compare to traditional model-based interactive topic modeling techniques. We cannot assume that we will have metadata available to build tandem anchors, but we use them here because they provide a high water mark without the variance introduced by study participants.

2.4.1 Experimental Setup

We use the well-known 20 Newsgroups dataset (20NEWS) used in previous interactive topic modeling work: 18,846 Usenet postings from 20 different newgroups in the early 1990s.[1] We remove the newsgroup headers from each message, which contain the newsgroup names, but otherwise left messages intact with any footers or quotes. We then remove stopwords and words which appear in fewer than 100 documents or more than 1,500 documents.

To seed the tandem anchors, we use the titles of newsgroups. To build each multiword anchor facet, we split the title on word boundaries and expand any abbreviations or acronyms.

[1] `http://qwone.com/~jason/20Newsgroups/`

For example, the newsgroup title 'comp.os.ms-windows.misc' becomes {"computer", "operating", "system", "microsoft", "windows", "miscellaneous"}. We do not fully specify the topic; the title gives some intuition, but the topic modeling algorithm must still recover the complete topic-word distributions. This is akin to knowing the names of the categories used but nothing else. Critically, the topic modeling algorithm has no knowledge of document-label relationships.

2.4.2 Experimental Results

Our first evaluation is a classification task to predict documents' newsgroup membership. Thus, we do not aim for state-of-the-art accuracy,[2] but the experiment shows title-based tandem anchors yield topics closer to the underlying classes than Gram-Schmidt anchors. After randomly splitting the data into test and training sets we learn topics from the test data using both the title-based tandem anchors and the Gram-Schmidt single word anchors.[3] For multiword anchors, we use each of the combiner functions from Section 2.3.2. The anchor algorithm only gives the topic-word distributions and not word-level topic assignments, so we infer token-level topic assignments using Latent Dirichlet Allocation [16] with *fixed* topics discovered by the anchor method. We use our own implementation of Gibbs sampling with fixed topics and a symmetric document-topic Dirichlet prior with concentration $\alpha = .01$. With fixed topics, inference is very fast and can be parallelized on a per-document basis. We then train a hinge-loss linear classifier on the newsgroup labels using Vowpal Wabbit[4] with topic-word pairs as features. Finally, we infer topic assignments in the test data and evaluate the classification using those topic-word features. For both training and test, we exclude words outside the LDA vocabulary.

[2]The best system would incorporate topic features with other features, making it harder to study and understand the topical trends in isolation.

[3]With fixed anchors and data the anchor algorithm is deterministic, so we use random splits instead of the standard train/test splits so that we can compute variance.

[4]`http://hunch.net/~vw/`

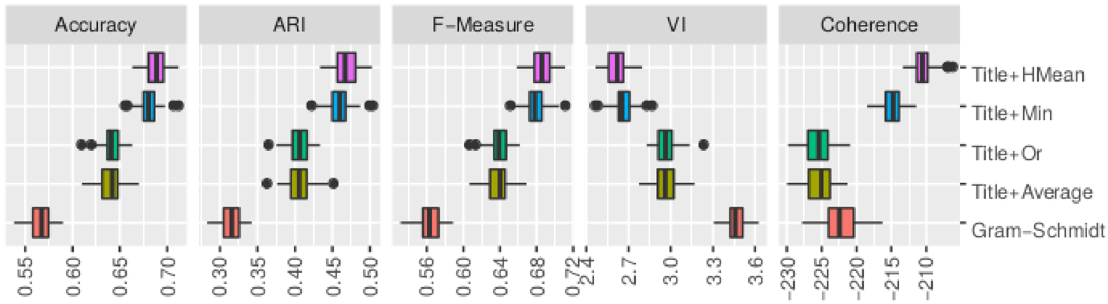


Figure 2.2: Using metadata can improve anchor-based topic models. For all metrics, the unsupervised Gram-Schmidt anchors do worse than creating anchors based on Newsgroup titles (for all metrics except VI, higher is better). For coherence, Gram-Schmidt does better than two functions for combining anchor words, but not the element-wise min or harmonic mean.

The topics created from multiword anchor facets are more accurate than Gram-Schmidt topics (Figure 2.2). This is true regardless of the combiner function. However, harmonic mean is more accurate than the other functions.[5]

Since 20NEWS has twenty classes, accuracy alone does not capture confusion between closely related newsgroups. For example, accuracy penalizes a classifier just as much for labeling a document from 'rec.sport.baseball' with 'rec.sport.hockey' as with 'alt.atheism' despite the similarity between sports newsgroups. Consequently, after building a confusion matrix between the predicted and true classes, external clustering metrics reveal confusion between classes.

The first clustering metric is the adjusted Rand index [119], which is akin to accuracy for clustering, as it gives the percentage of correct pairing decisions from a reference clustering. Adjusted Rand index (ARI) also accounts for chance groupings of documents. Next we use F-measure, which also considers pairwise groups, balancing the contribution of false negatives, but without the true negatives. Finally, we use variation of information (VI). This metric measures the amount of information lost by switching from the gold standard labels to the

[5]Significant at $p < 0.01/4$ when using two-tailed t-tests with a Bonferroni correction. For each of our evaluations, we verify the normality of our data [29] and use two-tailed t-tests with Bonferroni correction to determine whether the differences between the different methods are significant.

predicted labels [73]. Since we are measuring the amount of information lost, lower variation of information is better.

Based on these clustering metrics, tandem anchors can yield superior topics to those created using single word anchors (Figure 2.2). As with accuracy, this is true regardless of which combination function we use. Furthermore, harmonic mean produces the least confusion between classes.[6]

The final evaluation is topic coherence by Newman et al. [83], which measures whether the topics make sense, and correlates with human judgments of topic quality. Given V, the set of the n most probable words of a topic, coherence is

$$\sum_{v_1, v_2 \in V} log \frac{D(v_1, v_2) + \epsilon}{D(v_2)} \tag{2.7}$$

where $D(v_1, v_2)$ is the co-document frequency of word types v_1 and v_2, and $D(v_2)$ is the document frequency of word type v_2. A smoothing parameter ϵ prevents zero logarithms.

Figure 2.2 also shows topic coherence. Although title-based anchor facets produce better classification features, topics from Gram-Schmidt anchors have better coherence than title-based anchors with the vector average or the or-operator. However, when using the harmonic mean combiner, title-based anchors produce the most human interpretable topics.[6]

Harmonic mean beats other combiner functions because it is robust to ambiguous or irrelevant term cooccurrences in an anchor facet. Both the vector average and the or-operator are swayed by large outliers, making them sensitive to ambiguous terms in an anchor facet. Element-wise min also has this robustness, but harmonic mean is also able to better characterize anchor facets as it has more centralizing tendency than the min.

[6]Significant at $p < 0.01/4$ when using two-tailed t-tests with a Bonferroni correction. For each of our evaluations, we verify the normality of our data [29] and use two-tailed t-tests with Bonferroni correction to determine whether the differences between the different methods are significant.

2.4.3 Runtime Considerations

Tandem anchors will enable users to direct topic inference to improve topic quality. However, for the algorithm to be interactive we must also consider runtime. Cook and Thomas [28] argue that for interactive applications with user-initiated actions like ours the response time should be less than ten seconds. Longer waits can increase the cognitive load on the user and harm the user interaction.

Fortunately, the runtime of tandem anchors is amenable to interactive topic modeling. On 20NEWS, interactive updates take a median time of 2.13 seconds. This result was obtained using a single core of an AMD Phemon II X6 1090T processor. Furthermore, larger datasets typically have a sublinear increase in distinct word types, so we can expect to see similar run times, even on much larger datasets.

Compared to other interactive topic modeling algorithms, tandem anchors has a very attractive run time. For example, using an optimized version of the sampler for the Interactive Topic Model described by Hu and Boyd-Graber [50], and the recommended 30 iterations of sampling, the Interactive Topic Model updates with a median time of 24.8 seconds [50], which is well beyond our desired update time for interactive use and an order of magnitude slower than tandem anchors.

Another promising interactive topic modeling approach is Utopian [24], which uses non-negative factorization, albeit without the benefit of anchor words. Utopian is much slower than tandem anchors. Even on the small InfoVis-VAST dataset which contains only 515 documents, Utopian takes 48 seconds to converge. While the times are not strictly comparable due to differing datasets, Utopian scales linearly with the size of the data, we can intuit that even for moderately sized datasets such as 20NEWS, Utopian is infeasible for interactive topic modeling due to run time.

While each of these interactive topic modeling algorithms do achieve reasonable topics, only our algorithm fits the run time requirements for interactivity. Furthermore, since tandem

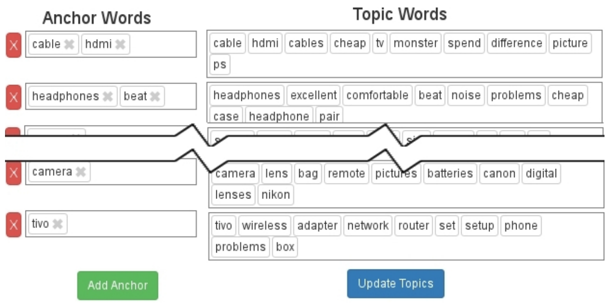


Figure 2.3: Interface for user study with multiword anchors applied to interactive topic modeling.

anchoring scales with the size of the vocabulary rather than the size of the data, this trend will only become more pronounced as we increase the amount of data.

2.5 Interactive Anchor Words

Given high quality anchor facets, the tandem anchor algorithm can produce high quality topic models (particularly when the harmonic mean combiner is used). Moreover, the tandem anchor algorithm is fast enough to be interactive (as opposed to model-based approaches such as the Interactive Topic Model). We now turn our attention to our main experiment: tandem anchors applied to the problem of interactive topic modeling. We compare both single word and tandem anchors in our study. We do not include the Interactive Topic Model or Utopian, as their run times are too slow for our users.

2.5.1 Interface and User Study

To show that interactive tandem anchor words are fast, effective, and intuitive, we ask users to understand a dataset using the anchor word algorithm. For this user study, we recruit twenty participants drawn from a university student body. The student median age is twenty-two. Seven are female, and thirteen are male. None of the students had any prior familiarity with topic modeling or the 20NEWS dataset.

Each participant sees a simple user interface (Figure 2.3) with each topic given as a row with two columns. The left column allows users to view and edit topics' anchor words;

the right column lists the most probable words in each topic.[7] The user can remove an anchor word or drag words from the topic word lists (right column) to become an anchor word. Users can also add additional topics by clicking the "Add Anchor" to create additional anchors. If the user wants to add a word to a tandem anchor set that does not appear in the interface, they manually type the word (restricted to the model's vocabulary). When the user wants to see the updated topics for their newly refined anchors, they click "Update Topics".

We give each participant a high level overview of topic modeling. We also describe common problems with topic models including intruding topic words, duplicate topics, and ambiguous topics. Users are instructed to use their best judgement to determine if topics are useful. The task is to edit the anchor words to improve the topics. We asked that users spend at least twenty minutes, but no more than thirty minutes. We repeat the task twice: once with tandem anchors, and once with single word anchors.[8]

2.5.2 Quantitative Results

We now validate our main result that for interactive topic modeling, tandem anchors yields better topics than single word anchors. Like our title-based experiments in Section 2.4, topics generated from users become features to train and test a classifier for the 20NEWS dataset. We choose this dataset for easier comparison with the Interactive Topic Modeling result of Hu et al. [52]. Based on our results with title-based anchors, we use the harmonic mean combiner in our analysis. As before, we report not only accuracy, but also multiple clustering metrics using the confusion matrix from the classification task. Finally, we report topic coherence.

Figure 2.4 summarizes the results of our quantitative evaluation. While we only compare user generated anchors in our analysis, we include the unsupervised Gram-Schmidt anchors as a baseline. Some of the data violate assumptions of normality. Therefore, we use

[7]While we use topics generated using harmonic mean for our final analysis, users were shown topics generated using the min combiner. However, this does not change our result.

[8]The order in which users complete these tasks is counter-balanced.

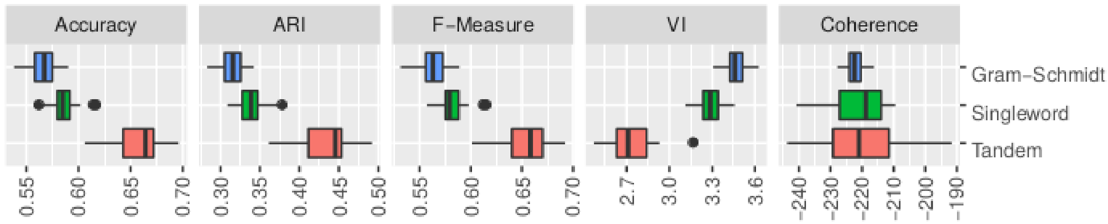


Figure 2.4: Classification accuracy and coherence using topic features gleaned from user provided multiword and single word anchors. Grahm-Schmidt anchors are provided as a baseline. For all metrics except VI, higher is better. Except for coherence, multiword anchors are best.

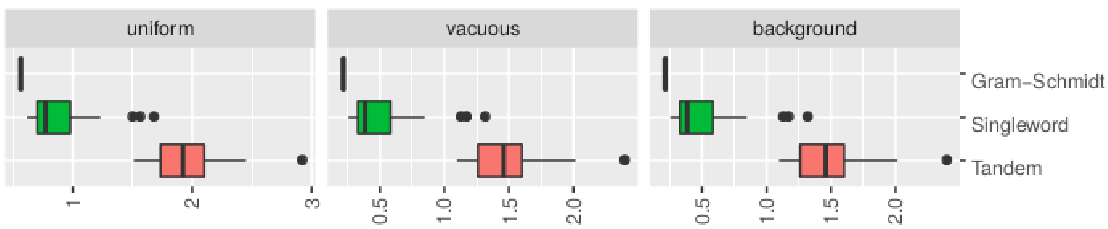


Figure 2.5: Topic significance for both single word and multiword anchors. In all cases higher is better. Multiword anchors produce topics which are more significant than single word anchors.

Wilcoxon's signed-rank test [117] to determine if the differences between multiword anchors and single word anchors are significant.

Topics from user generated multiword anchors yield higher classification accuracy (Figure 2.4). Not only is our approach more scalable than the Interactive Topic Model, but we also achieve higher classification accuracy than Hu et al. [52].[9] Tandem anchors also improve clustering metrics.[10]

While user selected tandem anchors produce better classification features than single word anchors, users selected single word anchors produce topics with similar topic coherence scores.[11]

[9]However, the values are not strictly comparable, as Hu et al. [52] use the standard chronological test/train fold, and we use random splits.

[10]Significant at $p < 0.01$ when using Wilcoxon's signed-rank test.

[11]The difference between coherence scores was *not* statistically significant using Wilcoxon's signed-rank test.

To understand this phenomenon, we use quality metrics [1] for ranking topics by their correspondence to genuine themes in the data. Significant topics are likely skewed towards a few related words, so we measure the distance of each topic-word distribution from the **uniform** distribution over words. Topics which are close to the underlying word distribution of the entire data are likely to be **vacuous**, so we also measure the distance of each topic-word distribution from the underlying word distribution. Finally, **background** topics are likely to appear in a wide range of documents, while meaningful topics will appear in a smaller subset of the data.

Figure 2.5 reports our topic significance findings. For all three significance metrics, multiword anchors produce more significant topics than single word anchors.[12] Topic coherence is based solely on the top n words of a topic, while both accuracy and topic significance depend on the entire topic-word distributions. With single word anchors, topics with good coherence may still be too general. Tandem anchors enables users to produce topics with more specific word distributions which are better features for classification.

2.5.3 Qualitative Results

We examine the qualitative differences between how users select multiword anchor facets versus single word anchors. Table 2.2 gives examples of topics generated using different anchor strategies. In a follow-up survey with our users, 75% find it easier to affect individual changes in the topics using tandem anchors compared to single word anchors. Users who prefer editing multiword anchors over single word anchors often report that multiword anchors make it easier to merge similar topics into a single focused topic by combining anchors. For example, by combining multiple words related to Christianity, users were able to create a topic which is highly specific, and differentiated from general religion themes which included terms about Atheism and Judaism.

[12]Significant at $p < 0.01$ when using Wilcoxon's signed-rank test.

Anchor	Top Words in Topic
Automatic Gram Schmidt	
love	love god evolution romans heard car
game	game games team hockey baseball heard
Interactive Single-word	
evolution	evolution theory science faith quote facts
religion	religion god government state jesus israel
baseball	baseball games players word teams car
hockey	hockey team play games season players
Interactive Tandem	
atheism god exists prove	god science evidence reason faith objective
christian jesus	jesus christian christ church bible christians
jew israel	israel jews jewish israeli state religion
baseball bat ball	hit baseball ball player games call
hockey nhl	team hockey player nhl win play

Table 2.2: Comparison of topics generated for 20NEWS using various types of anchor words. Users are able to combine words to create more specific topics with tandem anchors.

While users find that using tandem anchors is easier, only 55% of our users say that they prefer the final topics produced by tandem anchors compared to single word anchors. This is in harmony with our quantitative measurements of topic coherence, and may be the result of our stopping criteria: when users judged the topics to be useful.

However, 100% of our users feel that the topics created through interaction were better than those generated from Gram-Schmidt anchors. This was true regardless of whether we used tandem anchors or single word anchors.

Our participants also produce fewer topics when using multiword anchors. The mean difference between topics under single word anchors and multiple word anchors is 9.35. In follow up interviews, participants indicate that the easiest way to resolve an ambiguous topic with single word anchors was to create a new anchor for each of the ambiguous terms, thus explaining the proliferation of topics for single word anchors. In contrast, fixing an ambiguous tandem anchor is simple: users just add more terms to the anchor facet.

2.6 Conclusion

Tandem anchors extend the anchor words algorithm to allow multiple words to be combined into anchor facets. For interactive topic modeling, using anchor facets in place of single word anchors produces higher quality topic models and are more intuitive to use. Furthermore, our approach scales much better than existing interactive topic modeling techniques, allowing interactivity on large datasets for which interactivity was previous impossible.

Chapter 3

Labeled Anchors: a Scalable, Transparent, and Interactive Classifier

Published in Proceedings of the Conference on Empirical Methods in Natural Language Processing 2018 [67]

Abstract

We propose Labeled Anchors, an interactive and supervised topic model based on the anchor words algorithm [6]. Labeled Anchors is similar to Supervised Anchors [86] in that it extends the vector-space representation of words to include document labels. However, our formulation also admits a classifier which requires no training beyond inferring topics, which means our approach is also fast enough to be interactive. We run a small user study that demonstrates that untrained users can interactively update topics in order to improve classification accuracy.

3.1 Introduction

In this paper, we concern ourselves with the problem of interactive and transparent text classification. The value of such a classifier can be seen in the events shortly before the 2016 US presidential election when FBI Director James Comey notified Congress that the FBI had obtained emails from candidate Hillary Clinton's private email server which potentially contained state secrets. Nearly a week later, just two days before the election, Comey announced that nothing had been found in the emails that warranted prosecution. Many speculate that the timing of these announcements may have influenced the election.

There are times when the ability to quickly analyze large quantities of text is of critical importance. In the case of the Clinton emails, manual inspection appears to have been possible in one week's time, but there could have been less controversy if the emails had been categorized in a shorter period of time. Furthermore, in future cases the data may be too large for manual analysis.

While there are many text classification algorithms, none are both interactive and transparent at scale. We require interactivity because we would like to leverage human intuition to improve classification accuracy for a specific task. Transparency not only enables interactivity, but also allows users to inspect the classifier and gain confidence in the results.

Topic models such as Latent Dirichlet Allocation (or LDA) [16] aim to automatically distill large collections of documents into topics. These topics can be used to perform document classification [96]. Furthermore, work has been done to increase the human interpretability of topics [76]. Traditionally, topic models are graphical models which typically scale poorly to large data. A faster alternative is the Anchor Words algorithm, which relies on non-negative matrix factorization to infer topics [6]. Ordinarily, this factorization is NP-Hard [4], but with certain separability assumptions related to "anchor" words which uniquely identify topics, the factorization is scalable.

The Interactive Topic Model [52] allows human knowledge to be injected into the model in order to shape the topics in some meaningful way. While this model does incorporate user feedback, it is not fast enough to be truly interactive. A more scalable alternative is Tandem Anchors, which allows users to specify anchor words in order to influence the resulting topics [66].

A separate line of topic modeling research deals with supervised topic modeling, which allows document labels to influence topic inference [15]. The most recent work on supervised topic modeling is Supervised Anchors [86]. This approach uses document labels to influence the selection of anchor words, which in turn affects the resulting topics. However, Supervised Anchors requires a downstream classifier to be trained using topics as features.

Our main contribution combines the idea of Tandem Anchors with Supervised Anchors to produce text classification which is both interactive and transparent. Additionally, the mathematical approach we take to build this classification requires no training beyond inferring topics, unlike Supervised Anchors which requires both topic inference and significant additional time for training a downstream classifier. While Supervised Anchors requires the construction of an external classifier, our approach generates the classifier as part of topic inference. Consequently, our model is extremely fast and scalable compared to Supervised Anchors. We demonstrate that users are able to use our model to interactively improve document classification accuracy by manipulating topics.

3.2 Labeled Anchors

In this section we describe our approach, combining interactive and supervised topic modeling, which we call Labeled Anchors. We extend the Anchor Words algorithm [6] which takes as input a $V \times D$ matrix M of document-word counts and recovers a $V \times K$ matrix A of word probabilities conditioned by topic, where there are V word types, D documents, and K topics. Our approach extends this algorithm to incorporate L possible document labels.

3.2.1 Vanilla Anchor Words

In order to compute the topic-word matrix A, the Anchor Words algorithm uses a $V \times V$ cooccurrence matrix $\bar{Q}$. Each entry $\bar{Q}_{i,j}$ gives the conditional probability of word j occurring after observing word i in a document. Following Appendix D.1 of Arora et al. [6], $\bar{Q}$ is obtained by row-normalizing Q, which in turn is constructed using

$$Q = \bar{M}\bar{M}^T - \hat{M} \tag{3.1}$$

where $\bar{M}$ is a normalized version of the document-word matrix M giving equal weight to each document regardless of document length, and $\hat{M}$ accounts for words not cooccurring with themselves.

$\bar{Q}$ is a V-dimensional vector-space representation of each word and is used to compute a set of anchor words S. Each anchor word uniquely identifies a topic by having non-zero probability in one topic only. These anchors are computed using an adaptation of the Gram-Schmidt process from Arora et al. [6]. Once the set of anchor words S has been computed, we reconstruct the non-anchor words as a convex combination of the anchor word vectors. The coefficients of these combinations C are computed using exponentiated gradient descent to optimize

$$C_i = \underset{C_i}{argmin}\ D_{KL}(\bar{Q}_i || \sum_{k \in S} C_{i,k} \bar{Q}_k) \tag{3.2}$$

where i is the ith word of the vocabulary, $\bar{Q}_i$ is the vector-space representation of word i, and $D_{KL}(\cdot||\cdot)$ is Kullback-Leibler divergence.[1]

Because the occurrence pattern of each anchor word throughout the documents must mirror that of the topic it anchors, each coefficient $C_{i,j}$ gives the conditional probability of topic j occurring given word i. This is the inverse conditioning we desire in the topic-word matrix A. We can therefore compute A using Bayes' Rule by multiplying the coefficient matrix C with the empirical probability of each word to get the probability of a word given a particular topic.

3.2.2 Vector-Space Representations

Supervised Anchors [86] augments $\bar{Q}$ by appending L additional columns to $\bar{Q}$ corresponding to the probability of words cooccurring with the L possible document labels. Because this augmented vector-space representation includes dimensions corresponding to document labels, both the anchor words and the resulting topics will reflect the document labels.

[1] Alternatively, we can use l^2-norm in place of KL-divergence.

$$
\begin{bmatrix}
p(w_1 \mid w_1) & & \cdots & & p(\ell_1 \mid w_1) & \\
& \ddots & & & \vdots & \\
\vdots & & p(w_j \mid w_i) & & p(\ell_1 \mid w_i) & \cdots \\
& & & \ddots & \vdots & \\
p(w_1 \mid \ell_1) & \cdots & p(w_j \mid \ell_1) & \cdots & p(\ell_1 \mid \ell_1) & \\
& & \vdots & & & \ddots
\end{bmatrix}
$$

Figure 3.1: Labeled Anchors treats labels as observed words in each labeled documents and updates $\bar{Q}$ under this assumption, creating the additional rows and columns highlighted here.

Our algorithm, called Labeled Anchors, also augments the vector-space representation to include the L document labels. However, we do not directly modify $\bar{Q}$. Instead, we treat the L possible document labels as words and pretend that we observe these label pseudowords directly in each labeled document. [2] A graphical representation of this is shown in Figure 3.1.

Consequently, our document-word matrix M is a $(V + L) \times D$ matrix. The first V entries of each column of M give the word counts for a particular document. The last L entries are zero, except for the entry corresponding to the label of that document.

We then construct $\bar{Q}$ using Equation 3.1, obtaining an order $V + L$ square matrix. As with Supervised Anchors, these additional L dimensions guide anchor selection to include anchors which reflect the underlying document labels. When we use Equation 3.2 to compute C, we also obtain an additional L rows of coefficients which each correspond to the conditional probability of a topic given a label. Finally, the first V rows of A are computed using Bayes' Rule to give us the probability of words given topics.

Labeled Anchors inherits the run time characteristics of the original Anchor Words algorithm. As shown in Arora et al. [6], topic recovery requires $O(KV^2 + K^2VT)$, where V is the size of the vocabulary, K is the number of anchors/topics, and T is the average number of iterations (typically around 100 in our experiments). Since $V \gg K$, adding any modest number of topics (less than 200) does not noticeably increase the runtime. Furthermore, since

[2] We could add multiple such words per label, but our preliminary experiments indicate that one per label is sufficient.

vocabulary size tends to grow logarithmically with respect to the size of the data [46], this approach is scalable even for very large datasets.

3.2.3 Free Classifier

Note that once the cooccurrence matrix $\bar{Q}$ has been computed, the recovery of the topic-word matrix A scales with the size of the vocabulary, not the size of the data. However, Supervised Anchors requires topic assignments for each training document[3] for use as features for some downstream classifier. Therefore, the process of building a classifier scales linearly with the number of documents and can be time consuming compared to topic recovery.

In contrast, the formulation of Labeled Anchors allows us to construct a classifier with no additional training. To do so, rather than using LDA with fixed topics, we employ a simple model similar to Labeled LDA [91] with the following generative story for an individual document containing N words:

1. Draw label $\ell \sim Cat(\lambda)$
2. For each $i \in [1, ..., N]$:
    (a) Draw topic assignment $z_i|\ell \sim Cat(\psi_l)$
    (b) Draw word $w_i|z_i \sim Cat(\phi z_i)$

The prior over document labels λ is simply the proportion of each label in the training data. We can estimate topic-label probabilities ψ using the last L rows of the coefficient matrix C, while the word-topic probabilities ϕ are the first V rows of A. Using these

[3]This is typically done using LDA with fixed topics.

hyperparameters, we make predictions using the following:

$$\ell^* = \underset{\ell}{argmax}\ p(\ell|\mathbf{w}) = \underset{\ell}{argmax}\ p(\ell, \mathbf{w}) \tag{3.3}$$

$$= \underset{\ell}{argmax} \sum_{z_1=1}^{K} \dots \sum_{z_N=1}^{K} p(\ell, \mathbf{z}, \mathbf{w}) \tag{3.4}$$

$$= \underset{\ell}{argmax}\ p(\ell) \prod_{i=1}^{N} \sum_{z_i=1}^{K} p(z_i|\ell) p(w_i|z_i) \tag{3.5}$$

$$= \underset{\ell}{argmax}\ \lambda_\ell \prod_{i=1}^{N} \sum_{z_i=1}^{K} \psi_{\ell,z_i} \phi z_i, w_i \tag{3.6}$$

$$= \underset{\ell}{argmax}\ log\lambda_\ell + \sum_{i=1}^{N} log \left(\sum_{z_i=1}^{K} C_{\ell,z_i} A_{z_i,w_i} \right) \tag{3.7}$$

where Equation 3.4 unmarginalizes the probabilities across the word-topic assignments, Equation 3.5 uses the model's conditional independencies to expand and simplify the probabilities, Equation 3.6 explicitly uses the parameters from the generative model, and Equation 3.7 transitions to the matrix representations for these probabilities as found in Section 3.2.2. In Equation 3.7 we also switch to log space to mitigate numeric precision issues.

3.2.4 User Interaction

Assuming that $\bar{Q}$ is precomputed and fixed, Labeled Anchors is fast enough to allow interactive modification of the topics as well as interactive display of classification accuracy, even on large datasets. The final step to solving the problem of creating an interactive and transparent classifier is to allow users to inject domain specific knowledge into the topic model. To do so, we use the idea of Tandem Anchors [66], which allows users to manually select sets of words to form anchors.

Ordinarily, anchor words can be somewhat inscrutable to human users. Because anchor words must uniquely identify topics, good anchors are typically esoteric low-to-mid frequency words. Intuitive, high frequency words usually appear in multiple topics. However, if we examine Equation 3.2, we can see that the anchor words are just points in V-dimension

#Docs	#Vocab	Labeled	Supervised
39388	3406	.532s	17.1s
99955	4829	.886s	28.6s
990820	6648	1.10s	282s

Table 3.1: Runtime for Labeled Anchors and Supervised Anchors on various subsets of Amazon product reviews. Labeled Anchors is dramatically faster than Supervised Anchors and scales to much larger datasets.

space; they do not actually have to correspond to any particular word so long as that point in space uniquely identifies a topic.

Tandem Anchors allows multiple words to form a single anchor pseudoword by computing the element-wise harmonic mean of a set of words. Since the harmonic mean tends towards the lowest values, the resulting pseudoword anchor largely ignores superfluous cooccurrence patterns in the constituent words. Consequently, while individual words forming the anchor may be ambiguous, users can combine multiple ambiguous words to intuitively express a single coherent idea.

3.3 Experimental Results

Before running a user study to validate that Labeled Anchors works as an interactive and transparent classifier, we first run a synthetic experiment to determine the runtime characteristics of our algorithm. We take subsets of a large collection of Amazon reviews[4] to produce datasets of various sizes. Using this data, we compare the runtime of Labeled Anchors with that of Supervised Anchors. All results are obtained using a single core of an Intel Core i7-4770K.[5]

As shown in Table 3.1, Labeled Anchors is orders of magnitude faster than Supervised Anchors, even for moderately sized datasets. Both Labeled Anchors and Supervised Anchors require us to recover topic-word distributions, an operation which scales with the size of the vocabulary. However, Supervised Anchors also requires us to infer document-topic

[4] http://jmcauley.ucsd.edu/data/amazon

[5] Our Python implementation is available at `https://github.com/byu-aml-lab/ankura`.

distributions in order to train an external classifier, an operation which scales linearly with the number of documents. Since vocabulary size typically grows logarithmically with respect to the number of documents [46], Labeled Anchors scales much better than Supervised Anchors.

When a user updates the anchors, the system must reinfer the topics, create the classifier, and evaluate the development dataset, all within a few seconds. If the update is too slow, the interaction will suffer due to increased cognitive load on users [28]. Results from an exploratory user study confirm this: when participants are faced with update times around 10 seconds, they are not successful in their topic-based tasks.[6]

Having established that Labeled Anchors is fast enough to be interactive, we now demonstrate that participants can use our system to improve topics for classification. In order to demonstrate the role of human knowledge in interactive topic modeling, we ask the users to identify sentiment (i.e. product rating) rather than product category, since the natural topics which arise from the Anchor Words algorithm tend to reflect product category instead of rating. We preprocess a set of Amazon product reviews with standard tokenization, stopword removal, and by removing words which appear in fewer than 100 documents. After preprocessing, empty documents are discarded, resulting in 39,388 documents. We use an 80/20 train/test split, with 1,500 training documents reserved as development data. We recruit five participants drawn from a university student body. The median age is 21. Three are male and two are female. None of the students have any prior familiarity with topic modeling.

We present the participants with a user interface similar to that of Lund et al. [66][7]. Users can view and edit a list of anchor words (or rather, sets of words which form each anchor), and they can view the top ten most probable words for each topic. We display the classification accuracy on the development data to give users an indication of how they are doing. After a brief training on the interface, users are asked to modify the anchors to produce

[6] For this reason, we do not report interactive results with Supervised Anchors.
[7] Shown in Figure 2.3.

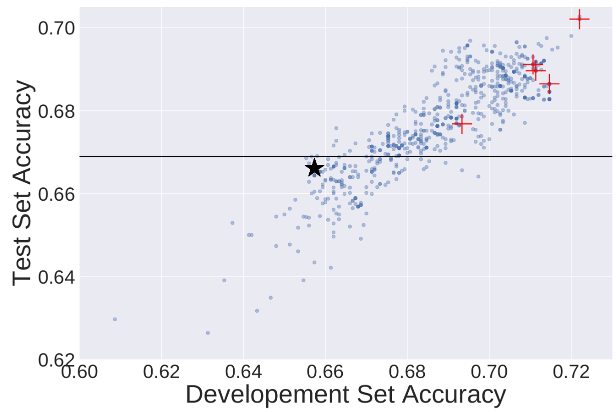


Figure 3.2: User study accuracy results comparing accuracy on the development set to the accuracy on the test set. The black horizontal line indicates the baseline accuracy from Supervised Anchors. The black star indicates the initial accuracy using Gram-Schmidt anchors with Labeled Anchors. The blue dots indicate various intermediate steps while editing the anchors. The red pluses are the final states after each user completes the task.

topics which reflect the underlying product ratings and improve the classification accuracy on the development dataset as much as possible. Participants are given forty minutes to perform this task.

Figure 3.2 summarizes the results of our user study. With just baseline anchors from Gram-Schmidt, the classification accuracy of Labeled Anchors is on par with that of Supervised Anchors using logistic regression as the downstream classifier. However, because Labeled Anchors is fast enough to allow interaction, participants are able to improve classification accuracy on the development set by an average of 5.31%. This corresponds to a 2.31% increase in accuracy on the test set.

We record each step of the user interactions and find a Pearson correlation coefficient of .88 between development accuracy and test accuracy. Thus, Labeled Anchors allows participants to interactively see updated classification accuracy and have confidence that held-out test accuracy will also improve.

With regard to the interaction that users had with the dataset, we observe several common strategies. Firstly, we notice that users who made more edits tend to have more success in terms of accuracy; this validates our assertion that slower update times hurt performance. Secondly, users end with a median of 21 topics, which is close to the 20 topics they start with, suggesting that either the users felt like this was an appropriate number of topics, or that they felt uneasy drastically changing the total number of topics from what they started with. Lastly, we find that users chose more single word anchors than we expected, with about 88% of anchors being single word anchors. However, we note that the multiword anchor which were used were typically short 2-3 word phrases which did not have an obvious single word counterpart, demonstrating that using tandem anchors did still give an appreciable benefit when combined with our labeled anchor approach.

3.4 Conclusion

Our results demonstrate that Labeled Anchors yields a classifier that is both human-interpretable and fast. Our approach not only combines the strengths of Supervised Anchors and Tandem Anchors, but introduces a mathematical construct for producing a classifier as a by-product of topic inference. Compared to Supervised Anchors, which requires costly training of a downstream classifier in addition to topic inference, our approach is much more scalable. Using Labeled Anchors, our participants are able to adjust the classifier so as to obtain superior classification results than those produced by Supervised Anchors alone.

Returning to our original motivating problem of quickly annotating a large collection of unlabeled emails, we assert that our approach could aid in quickly labeling the entire collection. With a modest investment of manual annotation, the initial training set could be labeled, and then with the help of our system the remaining documents could be automatically labeled in a transparent and explainable fashion.

Part II

Token Assignment

One advantage of anchor-based topic modeling is that we separate the problem of learning topic-word distributions from the problem of actually assigning individual topics to tokens. Having improved anchor selection in Part I, we are now prepared to explore methods of improving token-level topic assignment. In Chapter 4, we explore automated evaluation of local topic quality, giving us the tools to properly compare token-level topic assignment strategies. In Chapter 5 we use the result from Chapter 4 to experiment with popular methods of computing topic assignments. We experiment with both traditional coarse-grained topic models, as well as fine-grained topic models with large numbers of topics. We find that iterated conditional modes with a per-word initialization strategy works best in terms of local topic consistency, but that some downstream tasks such as topic-based classification are better served by mean-field variational inference with fixed topics. Our result in this chapter also validates the claim that fine-grained topic modeling is possible using anchor-based topic modeling.

Chapter 4

Automatic Evaluation of Local Topic Quality

To be submitted for publication at the Conference of the North American Chapter of the Association of Computational Linguistics 2019

Abstract

Topic models, even recent models which claim to improve the quality of token-level topic assignments, are typically evaluated with respect to the global topic-word distributions (e.g., coherence) without regard to local topic model quality found in the topic assignments. This can be problematic when topic assignments are used as features for downstream tasks such as document classification or when the assignments are shown to users as part of exploratory analysis. We propose a variety of automated measures of topic assignment quality, so that we may better understand and compare topic models at a local level. We also propose a new task designed to elicit human judgments of local topic quality. We correlate our proposed metrics with human evaluations on several datasets and find that a measure based on the number of times a document switches topics is effective. While the fact that minimizing topic switches agrees with human evaluations is intuitive, this result is surprising since our task only shows users individual topic assignments in isolation. We suggest that this new metric, which we call topic consistency, be adopted alongside global metrics such as topic coherence when evaluating new topic models.

4.1 Introduction

Topic models such as Latent Dirichlet Allocation (or LDA) [16] aim to automatically discover topics in a collection of documents, giving users a glimpse into the common themes of the data. Over the years, topic modeling literature has come to include a large number of different models, applied to a wide variety of different tasks.

Given the number of available topic models, for practitioners the question of model selection and model evaluation can be as daunting as it is important. In many cases, the question is easily answered when the model is used for a downstream task for which evaluation is possible (e.g., document classification). In most other cases, practitioners rely on automated metrics such as topic coherence [83] in order to compare topic model quality.

We review coherence and other existing metrics for topic model evaluation in Section 4.2. Generally speaking, these metrics evaluate topic models globally, meaning that the metrics evaluate the certain characteristics of the topics (i.e., distributions over corpus vocabulary) without regard to how the topics might be used locally.

However, in addition to producing global topic-word distributions, topic models also attribute each individual token to a specific topic. These local topic assignments are often used as features for downstream tasks. For example, topic-word pairs from the topic assignments might be used as features for a topic-based document classification system. If the topic assignments are inaccurate, the accuracy of the classifier may suffer.

Additionally, topic assignments may be used to facilitate exploratory analysis, in which humans examine topic assignments manually to make sense of the topical content of a document with respect to the rest of the corpus. While the global topic-word distributions generally make sense to a user and serve to give the user a high-level overview of the general themes and trends in the data, the quality of the local topic assignments can be bewildering to users.

For example, Figure 4.1 shows typical topic assignments using LDA. Arguably, most, or all, of the sentence should be assigned to the 'Music' topic since the sentence is about

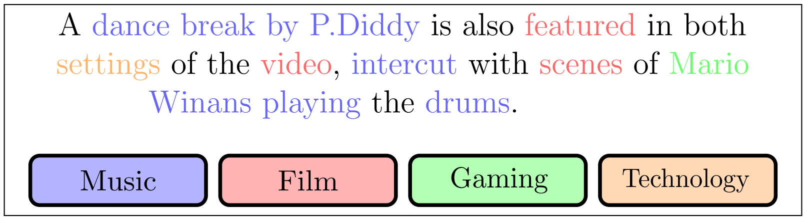


Figure 4.1: Topic assignments from LDA on a sentence from a Wikipedia document. Notice that even noun-phrases are split in a way which is bewildering to users.

a music video for a particular song, but parts of the sentence are assigned to other topics including 'Gaming' and 'Technology', possibly because other sentences in the same document are concerned with those topics. Even noun-phrases, such as 'Mario Winans' in Figure 4.1, which presumably should be assigned to the same topic, can be split across different topics.

This is important because accurate local topic assignments can help us classify and understand individual documents. However, if the accuracy of token-level topic assignments is poor, these downstream tasks and analysis may suffer even if the global topic-word distributions are coherent.

The problem of improving local topic assignments has received some attention from the modeling-side in topic modeling literature. For example, HMM-LDA [43] integrates syntax and topics by allowing words to be generated from a special syntax specific topic. TagLDA [126] adds a tag specific word distribution for each topic, allowing syntax to impose local topic structure. The syntactic topic model, or STM, extends this idea and generates topics using syntactic information from a parse tree. An alternative approach to improving local topic consistency is by adding a Markov property to topic assignments. The hidden topic Markov model (HTMM) does this by adding a switch variable on each word which determines whether to reuse the previous topic or generate a new topic. More recently, Balikas et al. [9] proposed SentenceLDA which assigns each sentence to a single topic. CopulaLDA [10] supersedes SentenceLDA, and instead uses copulas to impose topic consistency within each sentence of a document.

Despite the apparent interest in improving the quality of local topic assignments, topic models are still typically only evaluated with respect to global topic quality. Even the aforementioned models which aim to improve local quality are only evaluated globally.

This paper concerns itself with the evaluation of token-level topic assignment quality so that we may better understand which topic models produce good local topic quality for individual documents. Following the example of previous work on global topic evaluation [83], in Section 4.3 we first propose a variety of automated metrics designed to perform this evaluation. Then, in Section 4.4 we propose a user study designed to elicit human evaluations of local topic quality. We correlate those results with our proposed metrics in Section 4.5. In Section 4.6, we discuss these results, recommend a new metric of topic model evaluation, which we call consistency.

4.2 Global Evaluation

Early topic models such as LDA were typically evaluated using held-out likelihood or perplexity [16]. Wallach et al. [114] gives details on how to estimate these quantities. However, while held-out perplexity can be useful to test the generalization of predictive models, it has been shown to be negatively correlated with human evaluations of global topic quality [23]. These judgments were elicited using a topic-word intrusion task, in which human evaluators are shown the top n most probable words in a topic word distribution and asked to identify a randomly chosen 'intruder' word which was injected into the word list. This task operates under the assumption that if a topic is semantically coherent, then the intruder should be easy to identify.

While human evaluations of topic coherence are useful, automated evaluations are easier to deploy. Consequently, Newman et al. [83] proposed a variety of automated evaluations of topic coherence, and correlated these metrics with human evaluations with the topic-word intrusion task. They show that an evaluation based on aggregating pointwise mutual information (PMI) scores across the top n most likely terms in a topic distribution correlates well with human evaluations. We will follow a similar pattern in developing our own automated metrics for evaluating the quality of token level topic assignments.

The PMI scores can be computed with respect to the modeled data, but more frequently are computed using some large reference corpus such as Wikipedia. This metric, colloquially referred to simply as coherence, is currently the most popular form of automated topic model evaluation, although many variations exists, such as one measure which uses conditional probabilities of the top n instead of PMI scores [76]. Note that coherence is a measure of global topic quality, since it considers only the global topic-word distributions, without regard to the quality of the token-level topic assignments.

Topic coherence has been well studied. For example, through certain types of regularization, we can improve topic coherence [84]. Newman et al. [83] gave a methodology for automatically performing topic-word intrusion tasks directly. Since topic coherence depends on the choice of how many words from each topic to consider, work has been done exploring topic cardinality with respect to coherence [59].

Since topics are typically summarized by their top n most probable words, topic coherence is an important consideration when using topic modeling for the purpose of exploratory analysis. However, when topics are used as features for downstream tasks such as document classification, the characteristics of the entire topic-word distribution become more important.

Consider for example, two topics which rank the words of the vocabulary by probability in the same order. Suppose that one of these distributions was more uniform than the other (i.e., had higher entropy). While both distributions would give the same interpretation to a human examining the top n words of these topic-word distributions, the topic-word distribution with lower entropy places more weight on the high-rank words and is much more specific.

Using this intuition, AlSumait et al. [1] developed metrics for evaluating topic significance. While this work was originally used to rank topics by significance, it has been used to characterize entire models by measuring average significance across all topics in a single model [66].

Topic significance is evaluated by measuring the distance between topic distributions and some background distribution. For example, we can measure the distance of the topic-word distributions from the uniform distribution over words (SigUni). Alternatively, we can use the the empirical distribution of words in the corpus, or the vacuous distribution (SigVac) as our background distribution.

Like coherence, topic significance is a global measure of topic quality since it considers the topic distributions without regard to local topic assignments. However, it differs from topic coherence in that it considers the entire topic-word distribution. Lund et al. [66] found that when topics were used as features for document classification, models with similar coherence scores might perform differently on downstream classification accuracy, with better accuracy achieved by models with higher significance scores.

4.3 Proposed Metrics

As previously mentioned, recent models such as CopulaLDA [10], which claim to improve the quality of token-level topic assignments, have only been evaluated using global topic metrics. We develop an automated methodology for evaluating local topic model quality. We follow the pattern used by Newman et al. [83] to develop coherence and will propose a variety of potential metrics. As with coherence, we correlate these automated metrics with human evaluations in order to determine which automated metric yields the most accurate estimate of local topic quality as judged by human annotators.

Topic Switch Percent (SwitchP) Our first proposed metric measures local topic consistency. In a corpus with n tokens, with z_i being the topic assignment of the ith word in the corpus, and $\delta(i, j)$ being the Kronecker delta function, we measure this consistency with

$$\frac{1}{n-1}\sum_{i=1}^{n-1}\delta(z_i, z_{i+1}). \tag{4.1}$$

Essentially what this metric measures is the percent of times a topic switch occurs relative to the number of times it could have switched.

Topic Switch Variation of Information (SWITCHVI) Topic switch percent penalizes any topic switch without regard to how related the topics are. We utilize variation of information (or VI), which measures the amount of information lost in changing from one partition to another [73]. Assuming that our model has K topics, and once again using z_i as the topic assignment for word w_i, we consider two partitions $S = \{S_1, ..., S_K\}$ and $T = \{T_1, ..., T_K\}$ of the set of words w, such that $S_i = \{w_j | z_j = i\}$ and $T_i = \{w_j | z_{j+1} = i\}$. Variation of information is defined as

$$H(S) + H(T) - 2I(S, T) \tag{4.2}$$

where $H(\cdot)$ is entropy and $I(S, T)$ is the mutual information between S and T. In other words, we measure how much information we lose in our topic assignments if we reassign every word to the topic of the word that follows. Similar to topic switch percent, this measure penalizes topic switches. However, models which switch consistently between the same topics (presumably related topics) are penalized less than models which switch between topics at random.

Average Rank (AVGRANK) Even when evaluating local topic quality, the most common way of presenting topics to humans is as a set of related words, namely the most probable words in the topic-word distributions. Consequently, when human evaluators consider the quality of a topic assignment, they will still by influenced by the order or ranking of the words within a topic-word distribution, even if they are unaware of the actual probabilities. Leveraging this intuition, where $rank(w_i, z_i)$ is the rank of word i in topic i when sorted by probability, we use the following:

$$\frac{1}{n} \sum_{i=1}^{n} rank(w_i, z_i) \tag{4.3}$$

With this evaluation, lower is better. The lower bound is 1, although this would require that every word be assigned to a topic for which it is the mode, which is impossible unless the number of topics is equal to the vocabulary size.

Topic-Word Divergence (WordDiv) The previous metrics do not take into account the actual probabilities in the topic-word distributions. Suppose the topic-word distributions for a topic model with K topics, V token types, and D documents are given by a $K \times V$ matrix ϕ such that $\phi_{i,j}$ is the conditional probability of word j given topic i. Furthermore, let θ_d be the K-dimension document-topic distribution for the dth document (possibly obtained by normalizing the topic counts in z_d), and ψ_d be the V-dimensional distribution of words for document d (possibly obtained by normalizing the word counts in z_d). Our next metric measures how well the topic-word probabilities explain the words which are assigned to those topics:

$$\frac{1}{D}\sum_{d}^{D} JSD(\theta_d \cdot \phi || \psi_d) \tag{4.4}$$

where $JSD(P||Q)$ is the Jensen-Shannon divergence between the distributions P and Q. This formulation essentially rewards individual topic assignments which use topics which explain the cooccurrences of an entire document rather than individual tokens.

Window Probabilities (Window) Recalling the problem illustrated in Figure 4.1 in which groups of words such as noun phrases can be split across topics, our final proposed metric seeks to reward topic models which have topic assignments which not only explain individual words, but also the words within a window around the assignment. Given a window size s, and once again using ϕ as the topic-word distributions, we compute the following:

$$\frac{1}{n(2s+1)}\sum_{i}^{n}\sum_{j=i-s}^{i+s} \phi_{z_j,w_i} \,. \tag{4.5}$$

In our experiments, we use a window size of $s = 1$, meaning that for each word we consider its topic assignment, as well as the topic assignments for the words immediately preceding and following the initial word.

Dataset	Documents	Tokens	Vocabulary Size
Amazon	39388	1389171	3406
Newsgroups	18748	1045793	2578
New York Times	9997	2190595	3328

Table 4.1: Statistics on datasets used in user study and metric evaluation.

4.4 Human Evaluations

We follow the general design philosophy employed by Newman et al. [83] in developing the coherence metric. We learn a variety of models on various datasets in order to observe a wide range of token-level topic quality. We then evaluate these models using not only our proposed metrics, but using crowdsourcing data on a task designed to elicit human evaluation of local topic model quality. By correlating the human evaluation with automated metrics, we can determine how best to measure local topic quality. In this section, we first discuss the models we employ, then the crowdsourcing task. While not our main result, we also discuss the annotator agreement with each model type.

4.4.1 Datasets and Models

In order to correlate our automated metrics with human evaluations over a wide range of topic models, we choose three datasets from different domains and with different writing styles. These datasets include a collection of Amazon product reviews, the well known Twenty Newsgroups dataset, and a collection of news articles from the New York Times. Following best practices, we apply stopword removal using a standard list of stopwords. Additionally, we remove any token which does not appear in at least 100 documents. Statistics for these three datasets can be found in Table 4.1.

Once again aiming for a wide variety of topic models for our evaluation, for each of these datasets, we train three types of topic models. As a baseline, we train Latent Dirichlet Allocation [16] on each of the three datasets. Since CopulaLDA [10] is the most recent and reportedly the best model with respect to local topic quality, we also employ this model on each dataset. Finally, we use the Anchor Words algorithm [6], which is a fast and scalable alternative to traditional probabilistic topic models based on non-negative matrix factorization. By itself, Anchor Words only recovers the topic-word distributions, so we follow Nguyen et al. [86] and use variational inference for LDA with fixed topics to assign each word to a topic.

In addition to varying the datasets and model types, we train each model with a variety of different choices on the number of topics, with the hope of increasing the diversity of observed topic model quality. For both LDA and Anchor Words, we use 20, 50, 100, 150, and 200 for the number of topics. For CopulaLDA, we use 20, 50, and 100 for the number of topics[1]. We vary the number of topics to produce models with small numbers of coherent, albeit less significant, topics as well as models with large numbers of more significant topics, allowing us to observe user preferences on a variety of models. Since each model includes some amount of non-determinism, we trained five instances of each dataset, model and and topic cardinally and averaged our results.

In the interest of reproducibility, the data, the scripts for importing and preprocessing the data, and the code for training and evaluating these topic models are available in an open source repository[2].

4.4.2 Crowdsourcing Task

Our goal in designing a crowdsourcing task is to get human annotators to evaluate the quality of token-level topic assignments. Not only will this task allow us to evaluate and compare models, but it will allow us to develop automated metrics for evaluating local topic quality. However, this is a highly subjective question and inter-annotator agreement is an issue if we directly ask annotators to judge model quality. Instead, we prefer to ask users to perform a task which illuminates the underlying quality indirectly. It is for this reason that previous

[1]Unfortunately, CopulaLDA did not scale beyond 100 topics. In contrast to LDA and Anchor Words, which ran in minutes and seconds respectively, CopulaLDA took days to run using the authors' implementation. Our runs with 150 and 200 topics never finished, as they where finally killed due to excessive memory consumption.

[2]https://github.com/jefflund/ankura

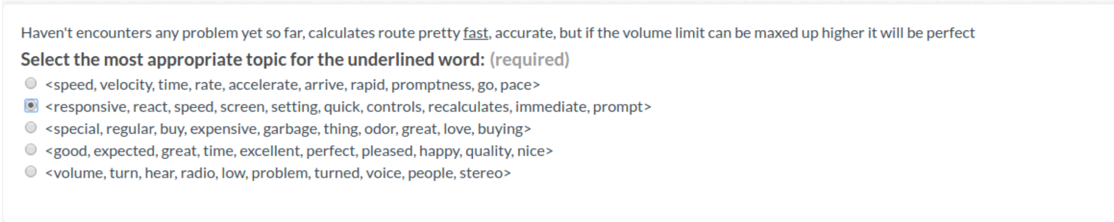


Figure 4.2: Example of the topic-word matching task. Users are asked to select the topic which best explains the underlined token.

crowdsourced evaluations of topic coherence relied on the word intrusion task instead of asking annotators to directly rate topic coherence [23].

In our proposed task, which we call topic-word matching, we show the annotator a short snippet from the data with a single token underlined. We show the user five topic summaries (i.e., the 10 most probable words in the topic-word distribution) and ask the user to select the topic which best fits the underlined token. One of the five options is the topic the model actually assigned to the underlined token, with the idea that the annotator will agree more often with a topic model which makes accurate local topic assignments. As alternatives to the model selected topic, we also include the three most probable topics in the entire document outside the topic the model selected for the underlined token. Since these topics likely appear in the same document, they may be somewhat related. However, a model which gives high quality token-level topic assignments should be able to consistently use the best possible topic for each individual token. That said, as seen in Figure 4.1, models such as LDA can struggle with this. Finally, we include a randomly selected intruder topic as a fifth option. This fifth option is included to help distinguish between an instance where the user sees equally reasonable topics for the underlined token (in which case, the intruding topic will not be selected), and when there are no reasonable options for the underlined token (in which case, all five topics are equally likely to be chosen). Figure 4.2 shows an example of this task shown to annotators.

For each of our 39 trained models (i.e., for each model type, dataset, and topic cardinality), we randomly select 1000 words to annotate. For each of the 39,000 selected words, we obtain 5 judgments. We aggregate the 5 judgments by selecting the contributor response with the highest confidence, with agreement weighted by contributor trust. Contributor trust is based on accuracy on test questions.

We deploy this task on a popular crowdsourcing website[3] and pay contributors $0.12 USD per page, with 10 annotations per page. For quality control on this task, each page contains one test question. The test questions in our initial piloted studies are questions we hand-select for their obvious nature. In the final study we use to obtain the results in Section 4.5, we also select a number of questions from the pilot studies with both high annotator confidence and perfect agreement to augment our bank of test questions. We require that contributors maintain at least a 70% accuracy on test questions throughout the job, and that they spend at least 30 seconds per page, but otherwise impose no other constraints on contributors.

4.4.3 Agreement Results

The main purpose of our user study is facilitate the exploration of automated evaluation techniques which correlate with human judgments of local topic quality. However, the results of our user study are interesting in their own right, so we briefly discuss them here.

We first measure inter-annotator agreement using Krippendorff's alpha with a nominal level of measurement [56]. Generally speaking, $\alpha = 1$ indicates perfect reliability, while $\alpha < 0$ indicates systematic disagreement. Over all the judgments we obtain, we compute a value of $\alpha = .44$. While this is nowhere near perfect agreement, this does indicate a moderate level of agreement.

Proper interpretation of this value is somewhat subjective. For comparison, we note the example given by Krippendorff [56] in which English-speaking annotators where tasked with assigning television characters to categories with Dutch names. On this task, annotators

[3]https://www.figure-eight.com

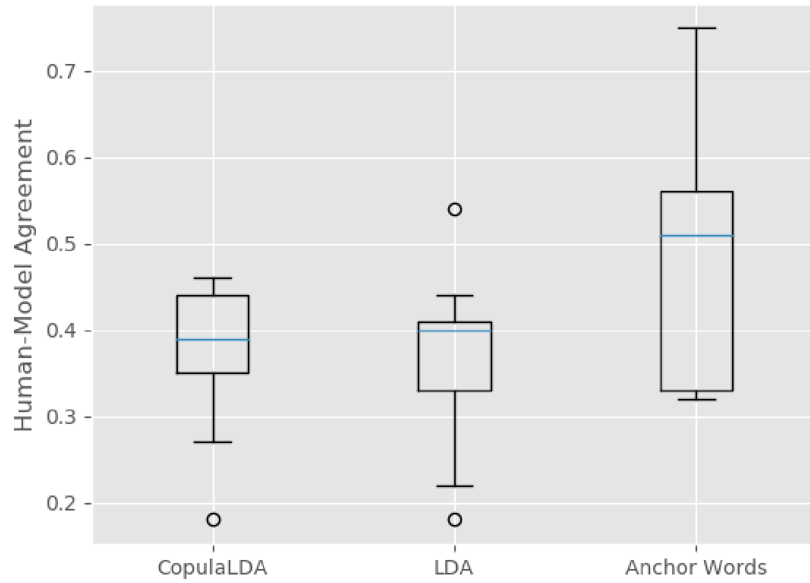


Figure 4.3: Plot showing human agreement with each model type. CopulaLDA performs slightly worse than LDA. Humans preferred topic assignments from Anchor Words by a wide margin.

also obtained $\alpha = .44$, which is coincidentally close to the value our annotators achieved on the topic-word matching task.

We note that when using crowdsourcing, particularly with subjective tasks such as topic-word matching, we expect somewhat low inter-annotator agreement. However, previous results indicate that when properly aggregated, we can still filter out noisy judgments and obtain reasonable opinions [87].

Our ultimate goal with using a wide variety of topic models was to obtain a wide range of local topic model quality so that we can determine which automated evaluation correleates best with human judgments. Figure 4.3 summarizes the human agreement with the three different model types. Surprisingly, despite claiming to produce superior local topic quality, and despite performing better than LDA in terms of perplexity (a global measure of topic quality), according to our results with the topic-word matching task, CopulaLDA actually performs slightly worse than LDA. We contend that this perfectly illustrates the need for automated metrics measuring local topic quality.

Equally surprising is the fact that users agree with Anchor Words more often than LDA by a wide margin, indicating that Anchor Words achieves superior token level topic assignments. However, in terms of global topic quality, Anchor Words is roughly similar to LDA [6]. One possible explanation is that Anchor Words separates the task of learning the global topic-word distributions from the problem of producing accurate local topic assignments, making both tasks easier. Of course, for many tasks an argument can be made for a joint-model, so further investigation into this phenomenon is warranted.

4.5 Automated Evaluations

We now turn our attention to the correlation between the human judgments obtained in Section 4.4.3 and the automated evaluations proposed in Section 4.3.

For each of our proposed metrics, we compute a least-squares regression for both the proposed metric and the human-model agreement on the topic-word matching task. As seen in Table 4.2, we report the coefficient of determination (r^2) for each metric and dataset.

	Metric	Amazon	Newsgroups	New York Times
Local Metrics	SwitchP	0.9077	0.8737	0.7022
	SwitchVI	0.8485	0.8181	0.6977
	AvgRank	0.5103	0.5089	0.4473
	WordDiv	0.3112	0.2197	0.0836
	Window	0.4884	0.3024	0.1127
Global Metrics	Coherence	0.4907	0.4463	0.3799
	SigUni	0.6310	0.4839	0.4935
	SigVac	0.6960	0.6081	0.6063

Table 4.2: Coefficient of determination (r^2) between automated metrics and crowdsourced topic-word matching annotations. We include metrics measuring both local topic quality such as switchp and measures of global topic quality such as coherence and significance.

Generally speaking, we find that humans agree more often with models trained on Amazon than on New York Times. This reflects the underlying data, since Amazon product reviews tend to be highly focused on specific products and product features and the topics naturally reflect those products. In contrast, New York Times data deals with a much wider array of subjects, and treats them with nuance and detail not found in typical product reviews, making the judgments more difficult and subjective.

Notwithstanding the differences across datasets, on all datasets, SwitchP most closely approximates human judgments of local topic quality, with an r^2 which indicates a strong correlation. This suggests that when humans examine token-level topic assignments, they are likely to value topic models which are locally consistent, meaning they are unlikely to switch topics from one token to the next. As evidenced by the lower r^2 for SwitchVI, even switching between related topics seems to contradict human judgments of local topic quality.

We also note that there is a correlation between coherence and the topic-word matching task, although the correlation is only moderate. Similarly, word-based significance metrics have a moderate correlation with topic-word matching. We maintain that these global topic metrics are important measures for topic model quality, but they fail to capture local topic quality as SwitchP does.

4.6 Discussion

Considering the intuition gained from the motivating example in Figure 4.1, it is not surprising that humans would prefer topic models which are locally consistent. Historically speaking, the fact that many topic models switch between topics so frequently is confusing to users. Thus, our result that SwitchP is correlated with human judgments of local topic quality is intuitive.

However, we note that our annotators were only shown single topic assignment in isolation. Since they were only shown the topic assignment for a single word, they could not have known whether the underlying model was locally consistent or not. Despite this, our annotators apparently preferred models which were consistent. Thus, while our result is intuitive, the fact that this result was illuminated through such an indirect measure is surprising.

Given our results, we recommend that topic switch percent be adopted as an automated metric to measure the quality of token-level topic assignments. We would refer to this metric colloquially as topic consistency in much the same way that PMI scores on the top n words of a topic are referred to as topic coherence. We advocate that future work on new topic models include validation with respect to topic consistency, just as most recent work has included evaluation of topic coherence.

However, we are careful to point out that topic consistency should not be used to the exclusion of other measures of topic model quality. After all, topic consistency is trivially maximized by simply minimizing topic switches without regard to the appropriateness of the topic assignment. Instead, we advocate that future models be evaluated with respect to global topic quality (e.g., coherence, significance, perplexity) as well as local topic quality (i.e., consistency). These measures, in addition to evaluation of applicable downstream tasks (e.g., classification accuracy) will give modelers and practitioners the information necessary to make informed decisions about model selection.

4.7 Conclusion

We develop a novel crowdsourcing task, which we call topic-word matching, to illicit human judgments of local topic model quality. Contrary to expectation, we find that CopulaLDA actually performs worse than other models with respect to this task. We apply this human evaluation to a wide variety of models, and find that topic switch percent or SwitchP correlates well with this human evaluation. We propose that this new metric, which we colloquially refer to as topic consistency, be adopted alongside measures of global topic quality for future work with topic model comparison.

Chapter 5

Token Level Topic Assignments

Abstract

Having established a method for evaluating topic consistency in Chapter 4, we now turn our attention to the problem of how best to produce token level topic assignments using fixed topics from the Anchor Algorithm. We experiment with both traditional coarse-grained topics models (e.g., a small number of topics) as well as fine-grained topic models which have large numbers of topics and try several assignment strategies including Gibbs sampling, mean field variational inference, and iterated conditional modes. We find that combining a per-word initialization with iterated conditional modes yields the best topic consistency, followed closely by mean-field variational inference. However, we also note that some tasks such as topic-based classification are best served by variational inference, albeit at the cost of a decrease in topic consistency.

5.1 Introduction

In Chapter 4, we established topic consistency (measured by the percent of words which switch topics from the preceding word) as an effective proxy for human evaluation of local topic quality. In this chapter, we build on this work to investigate the best way to produce token level topic assignments using fixed topics recovered by the Anchor Algorithm proposed by Arora et al. [6].

Under certain assumptions of separability [34], the anchor algorithm is able to provably recover the topic-word probabilities in polynomial time within certain bounds of accuracy [6].

However, recovery of the document-topic probabilities is expensive and uses a numerically unstable matrix inversion [4]. Consequently, practitioners utilizing the anchor algorithm typically infer per word topic assignments using Latent Dirichlet Allocation [16] with fixed topics[1] learned with the anchor algorithm. This inference is typically performed using mean field variational inference [86], since it has been shown to perform well for Latent Dirichlet Allocation in general [8].

However, other inference techniques have been used in conjunction with Latent Dirichlet Allocation. Since the problem of exact inference inference is NP-Hard [103], exact techniques such as expectation propagation [78] or belief propagation [89] have mostly been ruled out due to scalability issues. Instead, approximate inference algorithms such as expectation maximization [33] and Gibbs sampling [42] have been used. Gibbs sampling in particular is a very popular technique, since it is both easy to implement and performs well on several global measures of topic model quality. However, when paired with the hyperparameter optimization described by Wallach [112], variational inference performs as well as Gibbs sampling while being much more scalable [8].

More recent work has highlighted the potential usefulness of iterated conditional modes as an inference technique for topic models [65]. While iterated conditional modes has been known for some time [11], it has only recently received attention in topic modeling literature as a potential inference technique for facilitating interactive topic modeling. Fundamentally, iterated conditional modes is a hill climbing algorithm, so it will only find locally optimal solutions. Consequently, inference can be greatly affected by the initialization method.

Despite the wealth of literature exploring the full problem of inference for Latent Dirichlet Allocation and other similar models, no work has been done comparing inference techniques in the presence of fixed topics, much less topics obtained using the Anchor Words algorithm. This chapter explores this space and attempts to answer the question of the

[1] Latent Dirichlet Allocation with fixed topics is occasionally referred to as Latent Dirichlet Allocation with Static Topic-Word Distributions, or Explicit Dirichlet Allocation in the special case that the prior over the topic-word distributions is zero [45].

best method for making token level topic assignments. We first give a brief overview of the inference techniques we will explore. After introducing our experimental design, we then present our results in two parts: first for a typical number of topics, and then for a much higher number of topics suitable for fine-grained topic modeling.

5.2 Inference Techniques

Variational Inference When computing a *maximum a posteriori* estimate for a posterior of the form $p(\theta|w)$, variational inference seeks to minimize the Kullback-Leibler divergence between the true distribution p and an approximate but tractable distribution q [110]. In topic modeling literature, this approximate distribution is typically chosen using the mean field assumption, meaning that we fully factorize the exact distribution p as a product of marginals so that q has the form:

$$q(\theta) = \prod_i q_i(\theta_i). \tag{5.1}$$

Using the mean field assumption, we can perform updates that are algorithmically similar to expectation maximization to iteratively minimize the distance between p and q.

We note that the derivation of the variational updates can be difficult to produce, and many models do have published variation inference update equations. However, the updates for Latent Dirichlet Allocation are well known [16]. For our experiments, we utilize online stochastic variational updates [47] implemented by the popular Gensim package[2] modified to perform inference with fixed topics.

Gibbs Sampling Another popular technique for inference with Latent Dirichlet Allocation is Gibbs sampling [42]. A Markov chain is obtained by iteratively sampling the

[2]`https://radimrehurek.com/gensim`

topic assignment for each word according to the full conditional

$$p(z_{di}|z_{\neg di}, w) \propto (n_{d,z_i,\cdot} + \alpha) \frac{n_{\cdot,z_i,w_i} + \beta}{n_{\cdot,\cdot,w_i} + V\beta} \tag{5.2}$$

where z_{di} is the topic assignment for the ith word of the dth document, $z_{\neg di}$ is all of the topic assignments excluding the i word of the dth document, w is all the words of the corpus, $n_{d,t,v}$ is the count of words of type v assigned to topic t in document d, and dots represent marginalization over the replaced variable.

This method is straightforward to implement and has is available in a wide variety of software packages, including for the case when the topic-word distributions are fixed and represented by the topic-word counts $n_{\cdot,t,v}$. We use the implementation from the Ankura toolkit[3].

Iterated Conditional Modes Iterated conditional modes (or ICM) is a hill climbing technique related to Gibbs sampling. Rather than sampling a value for each latent variable, we instead maximize the full conditional. This maximization is performed iteratively on each latent variable until convergence. For Latent Dirichlet Allocation, this simply means applying the *argmax* to Equation 5.2. In our experiments, we use a modified version of the Gibbs sampler in the Ankura toolkit to perform iterated conditional modes.

While this technique has been known for some time [11], it has recently been given attention in topic modeling literature for the purpose of interactive topic modeling [65]. While iterated conditional modes is fast enough for interactive topic models, the algorithm is fundamentally coordinate-wise ascent, so some care must be taken with the choice of initialization. If a poor initialization strategy is used, then the algorithm is susceptible to poor local maxima. Consequently, in addition to reporting the results after a single run with uniform random initialization, we will also experiment with several initialization techniques:

Random Restart Typically, both Gibbs sampling and iterated conditional modes are initialized by assigning each word to a topic using a uniform distribution over topics. Uniform

[3]`https://github.com/jefflund/ankura`

random initialization does have problems with poor local maxima, but this can be mitigated to some degree by using random restart, or taking the maximum of some number of randomly chosen initializations. One advantage of this technique is that it is simple to implement and is embarrassingly parallel. In our preliminary experiments, we found that by taking the best of just 10 random restarts, we were able to significantly boost performance.

Per-Word Initialization Rather than initialize the per-word topic assignments using a uniform random distribution, an obvious strategy is to initialize each word using the topic that gives the most mass to the individual word. Because of polysemy, and the fact that Equation 5.2 encourages documents to use a small number of topics, this initialization will likely not be a local maximization, but it is more likely to at least start with promising topics.

5.3 Experimental Setup

In order to explore the space of token assignment strategies, we will utilize three different datasets, each with its own domain and writing style. These datasets include a collection of Amazon product reviews, the well known Twenty Newsgroups dataset, and a collection of news articles from the New York Times. On each of these datasets, we learn two sets of topics using the Anchor Words algorithm: one for standard coarse-grained topic modeling, and another with a larger number of topics for fine-grained topic modeling.

For both sets of topics, we apply standard tokenization and stopword removal. For the coarse-grained topics, we also apply standard vocabulary pruning, by removing any word which appears in fewer than 150 documents. For fine-grained topics, we need less aggressive vocabulary pruning so that there are enough terms in the vocabulary to allow a meaningful selection of a large number of anchor words. For our experiments with fine-grained topics, we use a value of 15 for this threshold.

We determine the number of topics to be used based on the number of documents in each dataset, with the intuition that the relationship between the number of documents and the number of topics determines the number of documents each topic must explain. For

Dataset	Documents	Vocabulary Size (Coarse/Fine)	#Topics (Coarse/Fine)
Amazon	39388	3406 / 12428	40 / 2000
Newsgroups	18748	2578 / 12422	20 / 1000
New York Times	9997	3328 / 18800	10 / 500

Table 5.1: Statistics on datasets and number of topics used in the study of token level topic assignments.

example, there are 18748 documents in the Twenty Newsgroups dataset. With a typical number of topics, say 100, each topic is responsible for roughly 187 documents, which is fairly coarse granularity. On the other hand, if we use 1000 topics, each topic must only explain an average of about 19 documents, which is much more fine- grained.[4] Table 5.1 summarizes the datasets and choices on the number of topics. Note that for the fine-grained topics, the threshold on how many documents a word must appear in before being considered as a candidate anchor must be decreased. For coarse-grained topics, we use the default of 500 documents, as per Arora et al. [6]. For fine-grained topics, we lower this threshold to 15 documents (matching the rare word threshold), since with a more standard threshold there are not enough words to form the number of anchors we need for each topic to explain a smaller number of documents.

For both topic-word distributions, we then employ each of the proposed assignment techniques discussed in Section 5.2. Since the topic-word distributions are fixed, global evaluations such as coherence are not useful for distinguishing between token assignment strategies. However, as discussed in Chapter 4, there are several ways we can evaluate the local topic quality. Most importantly, we will measure the local topic consistency by computing the percentage of words which share a topic its preceding word in a document. We also utilize a measure of topic significance introduced by AlSumait et al. [1], which computes the the distance of the uniform distribution over topics from the distribution of topics across documents.[5] Finally, we can use a downstream task, namely document classification, as a

[4]This assumes that each topic is used in roughly equal proportions, which is not necessarily the case depending on the assignment method.

[5]AlSumait et al. [1] also provide ways of measuring significance using topic-word distributions. However, these measures ignore the actual assignments and would be the same for each assignment strategy.

Dataset	Algorithm	Consistency (Coarse)	Consistency (Fine)
Newsgroups	Gibbs	0.026	0.001
	ICM+Single	0.705	0.187
	ICM+Restart	0.628	0.209
	ICM+Word Init	**0.709**	**0.265**
	Variational	0.553	0.250
Amazon	Gibbs	0.051	0.001
	ICM+Single	0.911	0.264
	ICM+Restart	0.851	0.266
	ICM+Word Init	**0.929**	**0.252**
	Variational	0.833	0.257
New York Times	Gibbs	0.102	0.002
	ICM+Single	0.978	**0.194**
	ICM+Restart	0.973	0.185
	ICM+Word Init	**0.981**	0.172
	Variational	0.929	0.133

Table 5.2: Topic consistency results. Higher is better. Bold indicates the best result across each dataset and topic set. Coarse topics results is more consistent models. However, regardless of topic granularity, ICM+Word Init yields the most locally consistent topic assignments except for New York Times with fine-grained topics.

proxy for local topic quality. We use a simple linear classifier with hinge loss[6]. We use the per-document topic assignments as features with which to train a classifier and then evaluate accuracy on a held-out test set. For Amazon, we predict product ratings. For Newsgroups, we predict the newsgroup from which each document originated. While our New York Times data is useful from the standpoint that comes from a drastically different domain than our other datasets, it contains no predictable metadata, so it will only be evaluated with respect to consistency and significance.

5.4 Results

Table 5.2 reports the topic consistency results as measured by the percentage of words which switch topics from the topic of the previous word. Unsurprisingly, when there are fewer topics, topic switches occur less often, since each topic must explain more words in the data. However, regardless of the topic granularity, the topic assignment strategy can have an impact

[6] http://hunch.net/~vw

Dataset	Algorithm	Significance (Coarse)	Significance (Fine)
Newsgroups	Gibbs	0.817	3.775
	ICM+Single	3.198	5.854
	ICM+Restart	3.112	5.948
	ICM+Word Init	**3.293**	**6.133**
	Variational	2.435	5.419
Amazon	Gibbs	0.553	3.656
	ICM+Single	2.846	5.567
	ICM+Restart	2.775	5.537
	ICM+Word Init	**2.951**	**6.361**
	Variational	2.302	4.664
New York Times	Gibbs	0.041	1.017
	ICM+Single	2.726	**3.789**
	ICM+Restart	2.682	3.736
	ICM+Word Init	**3.045**	3.459
	Variational	1.84	2.736

Table 5.3: Topic significance as measured by divergence from the background document-topic distribution. Higher significance values are better. Bold indicates the best result across dataset and topic sets. Iterated Conditional Modes (ICM) yields the most significant topics, except for New York Times with fine-grained topics.

on the local topic consistency. Notably, Gibbs sampling yields very inconsistent results, while iterated conditional modes is more locally consistent. Given that Gibbs sampling is competitive with variational inference with global measures such as perplexity, this result may seem surprising. However, Gibbs sampling is meant to explore the entire posterior distribution, and purposefully samples new values for topic assignments, meaning that it is locally inconsistent by design. Perhaps most interesting is the fact that while iterated conditional modes yields the most consistent result, no one initialization strategy dominates the others, meaning that maximizing the probability of the data does not necessarily correlate with a more locally consistent topic model.

The higher consistency of iterated conditional modes is explained by the topic significance results shown in Table 5.3. Here topic significance is measured by the Kullback-Leibler divergence of the distribution of topics across documents from the uniform distribution across documents, with the idea that a significant topic will appear in a small number of documents rather than appear in a wide variety of documents. From the topic significance results, we see

Dataset	Algorithm	Accuracy (Coarse)	Accuracy (Fine)
	Gibbs	59.9%	26.1%
	ICM+Single	49.3%	43.2%
Newsgroups	ICM+Restart	55.4%	54.5%
	ICM+Word Init	61.5%	81.2%
	Variational	65.3%	**83.7**%
	Gibbs	67.3%	57.7%
	ICM+Single	64.4%	60.7%
Amazon	ICM+Restart	66.3%	62.4%
	ICM+Word Init	66.3%	68.5%
	Variational	67.1%	**69.1**%

Table 5.4: Classification accuracy results. Bold indicates the best result across dataset. When paired with a good assignment strategy, fine-grained topics improves classification accuracy over standard coarse-grained topics.

that iterated conditional modes outperforms both Gibbs sampling and variational inference. Since each document is focused on fewer, more significant topics, it is not surprising that there were fewer topic switches when using iterated conditional modes.

Higher topic significance (at least with respect to document-topic distributions) matters because it allows a more nuanced view of individual documents. As an example, consider two sets of topics learned from the Twenty Newsgroups dataset which deal with religious issues shown in Table 5.5. For the coarse-grained topics, there are only three topics which deal with religion. These three topics do not directly correspond to the three religious newsgroups found in the data (namely, 'talk.religion.misc', 'alt.atheism', and 'soc.religion.christian'). On the other hand, a number of religious topics are found in the fine-grained topics (only a small number of of them are shown in Table 5.5). From the small selection of religious topics shown, we see a topic corresponding to a discussion of the origins of the Bible found on 'talk.religion.misc', a discussion of cults found in 'alt.atheism', and a discussion around the philosophical problem of evil found in both 'alt.atheism' and 'soc.religion.christian'.

The fine-grained topics are more nuanced and detailed. However, despite using the same set of fine-grained topics, the topic significance as measured by the document-topic distributions vary depending on the topic assignment strategy. Gibbs sampling, for example,

Granularity	Top Topic Words
Coarse	jesus christ christian paul christians religion christian government religious jewish bible read church book christianity
Fine	jesus christ heaven sin father religion religious cult atheist atheism god son lord faith peace bible translation accurate scripture authority christian faith church christians christianity accept creation nature evil choice ...

Table 5.5: Example of topics inferred from the 20 Newsgroups dataset. These particular topics are topics which deal with religious issues. For coarse-grained topics, all three religious topics are shown. For fine-grained, only a small sample of the religious topics are shown. Fine-grained topics are much more nuanced and detailed.

exhibits much lower topic significance, meaning that these nuanced topics are much more evenly spread out across documents, making it more difficult to find specific documents or passages which are closely related to a fine-grained topic. Recall that in this case, topic significance is measured using Kullback-Leibler divergence of the document-topic distribution from the uniform distribution over topics. Thus, we can interpret the significance numbers as information gain with respect to the uniform distribution and we can say that a good topic assignment strategy such as iterated conditional modes with per-word initialization covers nearly twice the information as a poor strategy like Gibbs sampling, despite both using the exact same set of topics. In other words, the nuanced topics provided by a large number of anchor words are much more useful when combined with an effective assignment strategy.

Another place where we can see the effects of topic significance is with respect to downstream classification accuracy. Table 5.4 shows these classification results. For both 20 Newsgroups and Amazon product reviews, we see that it is possible to achieve a significant boost in classification accuracy when making predictions using fine-grained topics as features. The boost in classification accuracy from variational inference suggests that not only does this strategy offer more topic significance than many of the alternatives, but the assignments it makes are more accurate than the competing assignment strategies.

5.5 Conclusion

This chapter provide two important contributions. First, we demonstrate that some topic assignment strategies are more effective than others. With regards to topic consistency and topic significance, iterated conditional modes with per-word initialization seems to do very well. With regards to topic-based classification accuracy, variational inference performs well. Gibbs sampling is a poor assignment strategy with respect to topic consistency despite its popularity with Latent Dirichlet Allocation.

Second, we demonstrate that when combined with an effective topic assignment strategy, we can greatly increase the number of topics in our topic model. These more fine-grained topics are much more nuanced than traditional coarse-grained topics, and can be useful for improving performance on downstream tasks such as topic-based classification.

We note that this is not possible with traditional probabilistic topic model using models such as LDA. Wallach et al. [113] demonstrated that with proper hyperparameter optimization, LDA tends to only use a small (less than 100) number of topics. The remain topics are simply not used by the model. This is likely because the conditional distributions for LDA exhibit a "rich get richer" characteristic where topics which have many assignments are more likely to be reused for subsequent assignments.

In contrast, our approach using many fine-grained topics separates the problem of learning the topic-word distributions from the task of making the token-level topic assignments. We suggest that this approach will open up new use cases for topic modeling which require more nuanced and accurate topics than traditional topic modeling approaches allow.

Part III

Application

Our main motivation and thesis statement throughout this dissertation has been that fine-grained topic modeling enables new topic-based applications which are not currently possible using using traditional coarse-grained topics. While there are many such applications, in Part III we focus on the problem of automatic generation of cross-references. This problem attempts to link individual passages of text to other topically relevant passages in the same corpus. In Chapter 6 we explore this application, combining results from previous chapters to develop a cost-effective system for producing new cross-reference resources. We claim that the success of this system compared to approaches utilizing coarse-grained topic models validates our thesis statement.

Chapter 6

Cross-Referencing Using Fine-Grained Topic Modeling

To be submitted for publication at the Conference of the North American Chapter of the Association of Computational Linguistics 2019

Abstract

Cross-referencing, which links passages of text to other related passages, can be a valuable study aid for facilitating comprehension of a text. However, because of the prohibitive cost of producing cross-reference resources, they only exist for the most well-studied texts (e.g., religious texts). We develop a topic-based system for automatically producing candidate cross-references which can be easily verified by human annotators. Our system utilizes fine-grained topic modeling with thousands of highly nuanced and specific topics to identify verse pairs which are topically related. We demonstrate that our system can be cost effective compared to having annotators acquire the expertise necessary to produce cross-reference resources unaided.

6.1 Introduction

Cross-references, or references from one part of a text to other parts of a text, can help to elaborate upon or clarify a particular passage of text, and can be a useful tool for deep understanding of a text. Cross-references can also be used to form a network structure which can be used to analyze the relational structure of a text. The existence of a thorough and complete cross-reference resource can facilitate better scholarship of a text and help readers to quickly find clarifying information. In contrast to a word concordance, which simply shows passages which share a common keyword, cross-references often include links which do not necessarily share the same keywords, but are still related in some important way.

However, the process of creating such a resource can be expensive and time consuming. For example, the LDS edition of the Bible[1] published by the Church of Jesus Christ of Latter-day Saints which added numerous cross-references and topic-based categories, took hundreds of volunteers thousands of hours over seven years to produce [2]. This process involved collecting more than 19,900 manually curated entries from volunteers, and then editing and refining those references with a small committee of experts down to a final cross-reference database containing 12,475 entries.

Compared to annotation tasks such as part-of-speech tagging, producing cross-reference annotations is a much more labor intensive task. This is because annotators must become intimately familiar with a text in order to note that a particular passage is related to another passage they happen to recall. This level of familiarity and expertise with a particular text is not typically found unless the annotator has spent a great deal of time studying and rereading the text. In contrast, for tasks such as tagging, annotators can be shown individual sentences, and the annotation can be made with no familiarity with outside passages. We can of course ask annotators to evaluate individual cross-references by showing them two short passages and asking them if the two passages are in fact related, but because the number of potential cross-references grows quadratically with the size of the data, we need a way to filter potential cross-references to a manageable size.

For this reason, expansive cross-references have only been produced for the most well-studied texts (e.g., religious texts). Other texts, such as academic textbooks may include indices or other similar references, but these tend to be sparse, focusing on a small number of keywords rather than linking each individual passage with other relevant passages. Our goal in this chapter is to produce a system which utilizes fine-grained topic modeling in order to

[1] https://www.lds.org/scriptures/bible

dramatically lower the cost of producing a cross-reference resource for new texts. We do not expect that such a system will be perfect, but we hope that the system could be accurate enough to allow for annotators to simply review the proposed cross-references and associated passages without having to deeply understand the entire text.

6.2 Methodology

In this section, we describe our experimental setup, and how we approach the problem of automatically cross-reference generation.

6.2.1 Cross-reference Datasets

While exact statistics are impossible to obtain, the number of printed copies of the Bible is estimated to be more than 5 billion [44]. Because of the way Christians think about this text, it is perhaps the most well-studied text of all time, and one of the few texts in the world with extensive cross-reference data. While we aim to produce a method which works generally, we utilize the English Standard Version of the Holy Bible [12], since we have no ground truth for other secular texts in order to properly validate our method. We use this specific translation of the Bible because it is used on `openbible.info`. While our work focuses on this specific religious text, we hope that our work can be applied in the future to other texts, including literary classics and collections of historical documents.

To aid any readers who are unfamiliar with this religious text, we note that the term 'verse' refers to a short division of a chapter. The entirety of the text is divided up into books. Typically individual verses are referenced by the name of the book, the chapter number, and the verse number, e.g., Isaiah 25:8. For convenience, each verse reference in electronic versions of this paper is a hyperlink to `openbible.info`, showing the verse along with associated cross-references. For the sake of narrowing our focus, our efforts in cross-referencing the Bible will focus on finding topically related verses, though other work could potentially link larger passages, such as entire chapters or passages spanning multiple verses.

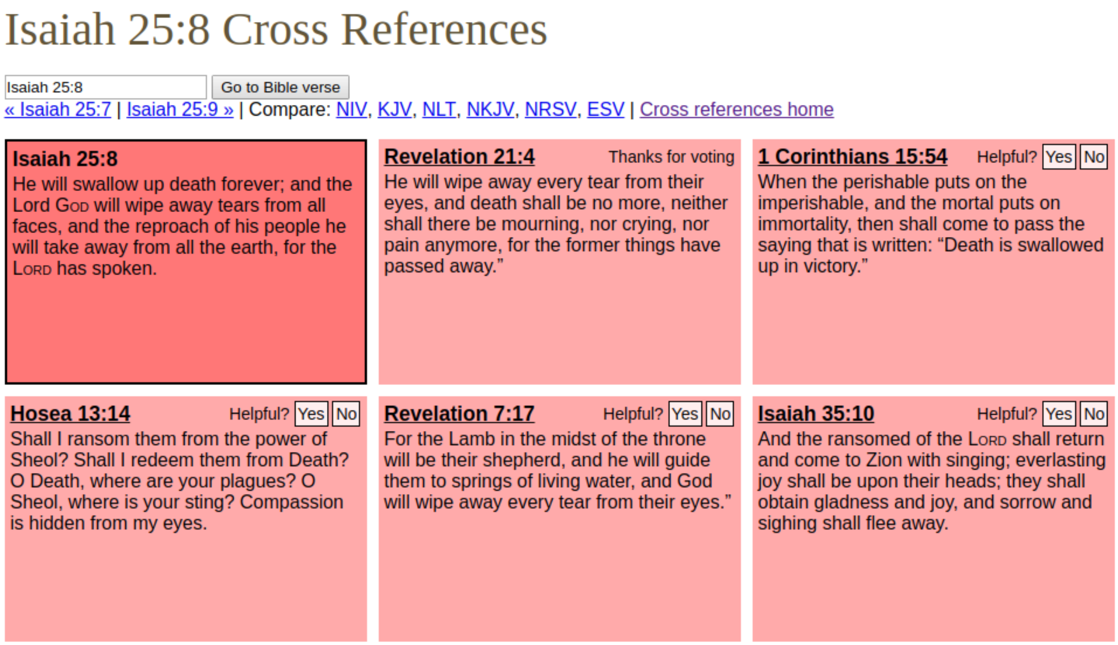


Figure 6.1: Voting interface for cross-reference data from `openbible.info`.

As a ground truth for cross-references, we utilize two sources. The first is the "Treasury of Scripture Knowledge, Enhanced" [80] (an extended version of the original "Treasury of Scripture Knowledge" [107]), which includes 670,796 cross-references between the 31,085 verses of the Bible. To our knowledge, this is the most exhaustive resource of human curated cross-references for the Bible to date. We will denote this cross-reference dataset as *TSKE*.

The second source of ground truth cross-references is a dataset from `openbible.info`. This dataset was seeded with various public domain cross-reference data, including the Treasury of Scripture Knowledge. As shown in Figure 6.1, users search for a verse they are interested in, and can then vote on whether they found a particular cross-reference to be helpful or not. With each helpful or not helpful vote counting as +1 and -1 respectively, the dataset includes the net result of the votes for each included cross-reference.

Thus, we can filter the dataset of cross-references based on how helpful each verse was rated to be. Counting only those cross-references which have a non-negative vote total, this dataset contains 344,441 cross-references. In figures, we denote this subset of the `openbible.net` cross-reference dataset as *OpenBible+0*. We also use the subset of cross-references which received a net total of 5 helpful votes. This subset, denoted as *OpenBible+5*, has 50,098 cross-references.

We do note however that the voting data does have some skewness in the number of votes for each cross-reference. The overwhelming majority of cross-references received fewer than five total votes for or against the reference. A small number of verses, including both popular verses as well as verses which happen to come from the very beginning of the Bible, can receive hundreds of votes.

6.2.2 Baselines for Automated Cross-reference Generation

These cross-reference datasets where produced at a tremendous cost in time and human efforts. To the best of our knowledge, efforts at automating this process are limited and have not received much attention in computer science literature. That said, a reasonable baseline for automated efforts is a simple word-based concordance, which lists words along with references to where the words occur in the text[2].

Using the TSKE as the ground truth for cross-references, this simple baseline will recover roughly 65% of the cross-references. For example, as shown in Figure 6.1 and assuming that stemming is performed, the verse Isaiah 25:8 would be properly linked to 1 Corinthians 15:54 due to the two verses sharing the terms 'death' and 'swallow'. On the other hand, verses such as Hosea 13:14 or 1 Corinthians 15:55 which reference the 'sting of death' in Isaiah 25:8 should not be linked to verses such as Revelation 9:10, which references the sting of a scorpion. For this reason, roughly 99% of cross-references found using word-based concordance are spurious according to the Treasury of Scripture Knowledge, making this baseline less useful as a cross-referencing resource.[3] We refer to this baseline as *word match*.

As a slightly stronger baseline, we also consider a topical concordance in which verses assigned to the same topic by some topic model are considered to be linked. We refer to this

[2]As an example of such a concordance for the King James Version of the Bible, see Strong's Concordance [104].

[3]For this reason, published biblical word concordances typically only give a manually curated subset of significant vocabulary terms.

baseline as *topic match*. For example, suppose that a topic model included a topic which gives high probability to terms such as 'death', 'swallow', 'victory' and 'sting'. Assuming such a model would make topic assignments associating the previously mentioned verse with this topic, then verses such as Isaiah 25:8, Hosea 13:14, and 1 Corinthians 14:54 would be linked, but Revelation 9:10 which uses the term 'sting' in a different context (i.e., the sting of a scorpion) would not likely be linked.

We can further increase the precision of this baseline by only linking references which share a topic and a word, although this does come at the cost of recall. We refer to this final baseline method as *topic-word match*[4].

6.2.3 Topic-Based Cross-Referencing

We now describe our approach to topic-based cross-referencing. The baseline approaches based on building word or topic concordances simply propose any cross-reference for which a word or topic matches another verse, meaning that we cannot set a threshold on the quality of the proposed cross-references. Instead, we propose comparing document-topic distributions as K-dimensional vectors, where K is the number of topics, and using standard vector distance metrics to compare verses. This idea has been used before [108], although not for the task of producing cross-references. By using a vector distance metric to compare the topical similarity of verse pairs, we can set a threshold on the number of proposed cross-references and propose only the most topically related verse pairs as cross-references. We experiment with four distance metrics: cosine distance, Euclidean distance, cityblock (or Manhattan) distance, and Chebyshev distance. However, given that previous work comparing document-topic vectors from LDA seem to default to cosine similarity [27, 108], we expect that cosine distance will be the best metric for selecting cross-references.

[4] Note that this is *not* the same thing as the topic-word matching task described in Chapter 4.

6.2.4 Model Selection

We claim that a fine-grained topic model, i.e., a topic model with a large number of highly nuanced topics, will be able to provide more value for tasks like cross-referencing than traditional coarse-grained topic models. In order to validate this claim, we will compare our fine-grained models with topics from a traditional Latent Dirichlet Allocation mode with 100 topics. We refer to this baseline model as *coarse*.

As discussed in previous chapters, traditional probabilistic topic models such as Latent Dirichlet Allocation are not able to utilize large numbers of topics. However, as demonstrated in Chapter 5, anchor-based topic models can be successfully trained with thousands of topics. Consequently, for our fine-grained models, we will employ the anchor word algorithm. Anchor-based topic models view topic modeling as non-negative matrix factorization. This class of topic models attempts to decompose a document-word matrix into two matrices, including a topic-word matrix which gives the conditional probabilities of a particular word given a topic. Ordinarily, this factorization is NP-Hard. However, given a set of anchor-words, or words which uniquely identify a topic, the computation requires only $O(KV^2 + K^2VI)$ where K is the number of topics, V is the size of the vocabulary, and I is the average number of iterations (typically around 100).

We train our anchor-based model using 3,000 topics. We choose this number based on the number of documents we expect each topic to explain: there are roughly 30,000 verses and, according to `openbible.info`, a median of 10 cross-references per verse, so we want each topic to be roughly responsible for 10 verses.

By default, the anchors for the 3,000 topics are produced using a modified form of the Gram-Schmidt process [6]. This process essentially views each word as a vector in high-dimensional space, and attemts to pick anchor words which maximally span that space. For more details, see Arora et al. [6]. In our results and figures, we refer to this model with the default anchor selection method as *Gram-Schmidt*.

This does present us with some difficulty with anchor words as this process tends to select the most extreme and esoteric anchors possible [60], which can lead to less useful topics as we increase the number of topics. However, in Chapter 2 we introduce a method of using multiple words to form a single anchor. This method, called tandem anchoring, was originally formulated as a way to extend the anchor algorithm to allow for interactive topic modeling [66]. However, in the case of fine-grained topic modeling, the algorithm is not scalable enough to allow interaction with such a large number of topics.[5]

Instead of utilizing human interaction to seed the topic anchors, we will seed the tandem anchors using the terms from randomly selected verses. For example, suppose we had randomly selected Isaiah 25:8 as a verse from which to form an anchor. As shown in Figure 6.1, this verse includes terms such as 'swallow', 'death', and 'tear'. Each of these terms is represented as a point in high-dimensional space. To produce a new point which represents our anchor, we simply average[6] the words forming the anchor. We repeat this process 3,000 times to produce an anchor-based topic model with tandem anchors. While this exact methodology of seeding topic anchors using randomly selected verses is novel, we note the similarity to the method used to seed topics in Rephil, a web scale topic model based used by Google [81]. In figures, we refer to this model with tandem anchors as *tandem*.

For each of these models, we must take the topic-word distributions from the topic model and produce document specific topic assignments. Drawing from the main result of Chapter 5, we utilize mean field variational inference in order to assign the individual verses to topics. As with our previous experiments with topic-based classification, our preliminary experiments with cross-referencing indicate that variational is nearly as good as iterated conditional modes with respect to topic consistency, and it performs slightly better with respect to cross-reference generation.

[5]With a typical number of topics, say 50, it takes less than 3 seconds to update on an AMD Phemon II X6 1090T processor. In contrast, with 3,000 topics, topic recovery takes approximately 2 hours and 20 minutes. This can be improved to a few minutes with proper parallelization (see Chapter 7), but even this is not fast enough for an interactive system.

[6]Specifically, we use the element-wise harmonic mean to average the words. See Chapter 2 for more details.

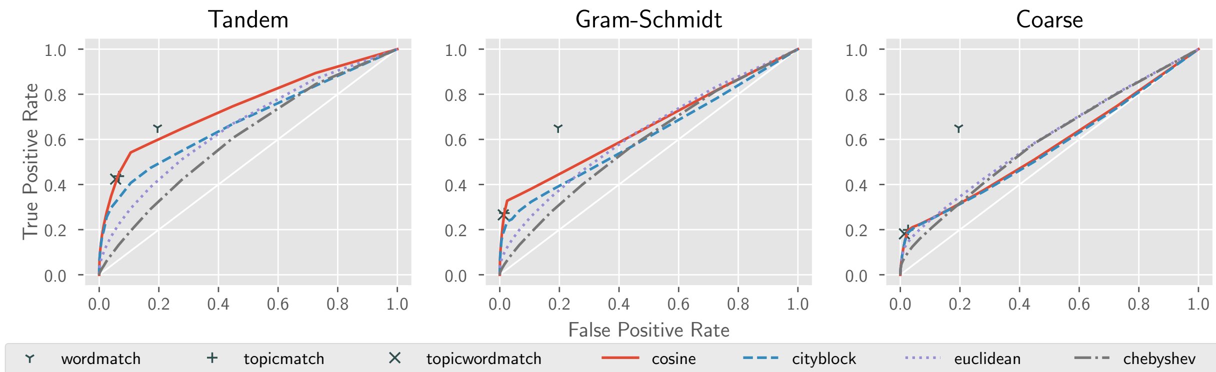


Figure 6.2: Cross-reference ROC curves for different metrics for cross-reference selection with topics from three different topic models and TSKE as the cross-reference ground-truth.

6.3 Results

In this section we present the results of our experiments with topic-based cross-referencing. As discussed in Section 6.2, we experiment with three different cross-reference datasets: TSKE, OpenBible+0, and OpenBible+5. We utilize three different topic models: coarse, Gram-Schmidt, and tandem, with the latter two being fine-grained topic models. We seek to demonstrate that our topic-based cross-referencing system can effectively utilize fine-grained topic modeling to produce candidate cross-references which can be annotated by humans in a cost-effective manner.

6.3.1 Metric Comparisons

We first explore the various metrics for selecting cross-references discussed in Section 6.2.3. With each proposed distance metric, we are able to set a threshold and determine which Bible verse pairs to keep as candidate cross-references and which to discard. Figure 6.2 and Figure 6.3 summarize our results with respect to metrics.

Figure 6.2 gives a receiver operator characteristic (or ROC) curve, which compares the true positive rate (or recall) against the false positive rate (or fall-out). We show the curve for the various metrics using the Treasury of Scripture Knowledge, Enhanced (or TSKE) as

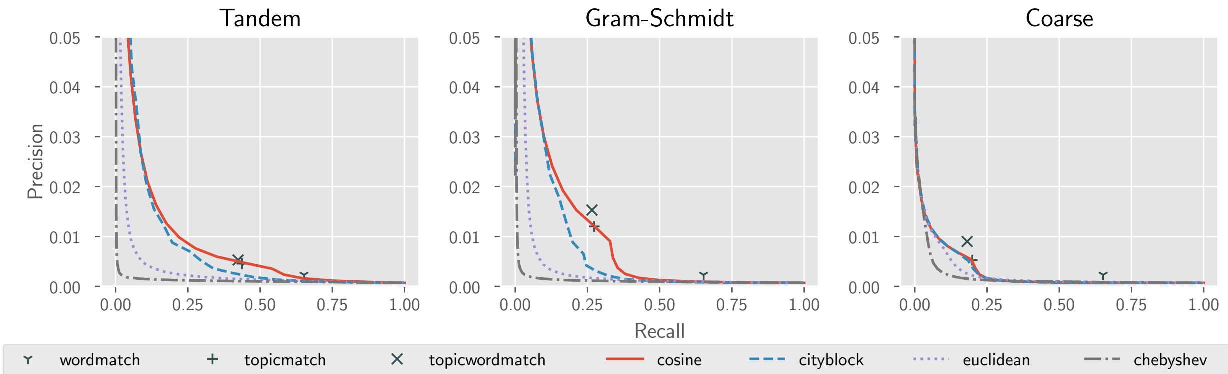


Figure 6.3: Cross-reference PRC plots with different metrics for cross-reference selection with topics from three different topic models and TSKE as the cross-reference ground-truth.

our ground truth. We show these curves on each of the three different models discussed in Section 6.2.4.

Overall, cosine distance is the best method for selecting cross-references, as it gives the largest area under the ROC curve. The major exception to this is with the traditional coarse-grained topic model for which Euclidean distance performs the best once the false positive rate is above 0.1. Considering that cosine distance has frequently been used in conjunction with topics from LDA [27, 108], this result is somewhat surprising.

Also of interest is the fact that the word-match baseline does reasonably well with respect to the true positive rate, at least if a false positive rate of roughly .196 is acceptable. Note that this corresponds to 188,974,806 false positives in the TSKE dataset, so while the 435,159 of true positives may be impressive, this baseline is not likely to be useful in practice, since it would require manual evaluation of such a large number of predicted positives. Indeed, due to the cost of manual evaluation, we are most interested in the part of the ROC curve which corresponds to a relatively small number of predicted positives.

While the receiver operator characteristic plot is undoubtedly more popular than the precision-recall plot, in cases where the data is imbalanced, the precision-recall plot can be much more informative. This is mainly because of the use of false-negatives in the ROC curve, which can present an overly optimistic picture of the classifier performance [97]. Consequently, in Figure 6.3, we also compare each of the proposed metrics using a precision-recall curve (or PRC).

The PRC plot reinforces the claim that cosine distance is the best distance metric to threshold cross-references since for any reasonable level of precision, cosine distance yields the best results, although for low recall rates, cityblock distance is competitive with cosine distance. The PRC plot is also a good illustration of why the matching baselines are not practically useful—they do have decent true positive rates, but the precision with these baselines is extremely low.

6.3.2 Topic Model Comparison

We now explore the various topic models discussed in Section 6.2.4. Figure 6.4 and Figure 6.5 summarize these results. Based on Section 6.3.1, each reported result in this section uses cosine distance to determine verse pairs which should be considered as candidate cross-references. We compare the results of the three topic models on the three datasets discussed in Section 6.2.1.

As shown in Figure 6.4, the overall trend on each of the three datasets is that the true positive rate increases as we use more selective datasets, likely because the more selective datasets like OpenBible+5 include fewer cross-references which are non-obvious or esoteric. However, as seen in Figure 6.5, on more selective datasets, we also see a drop in precision. Considering that OpenBible+5 is a subset of OpenBible+0, and that OpenBible+0 is nearly a subset of TSKE[7], this result is not surprising. In the final cost analysis (see Section 6.3.3), the selectivity of the actual annotators must be taken into account when attempting to predict the true positive rate or the precision of a system.

As shown in Figure 6.4, regardless of the anchor selection strategy, both fine-grained topic models outperform the traditional coarse-grained model, regardless of which dataset is used. Our model using tandem anchors built from randomly selected verses performs the

[7] OpenBible+0 has 533 references which are not included in TSKE seed from other public domain sources.

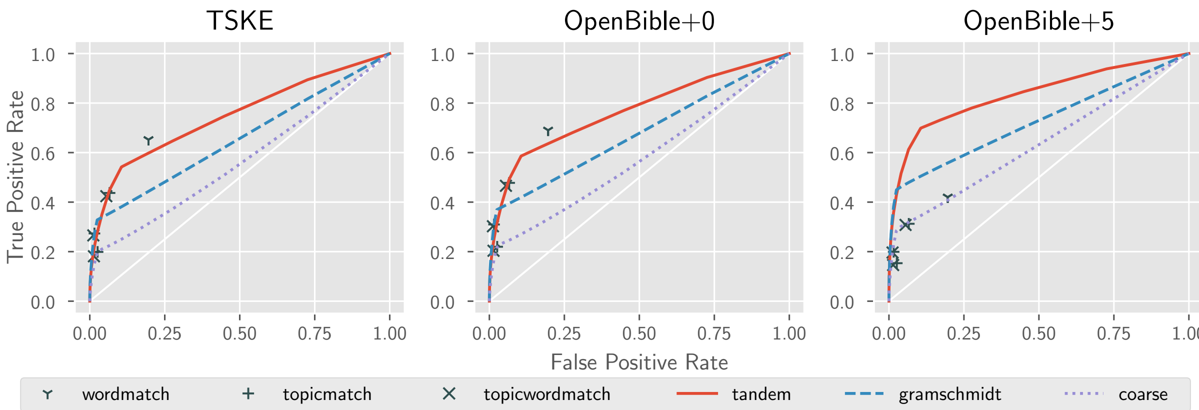


Figure 6.4: Cross-reference ROC curves with different models for cross-reference selection with three different datasets.

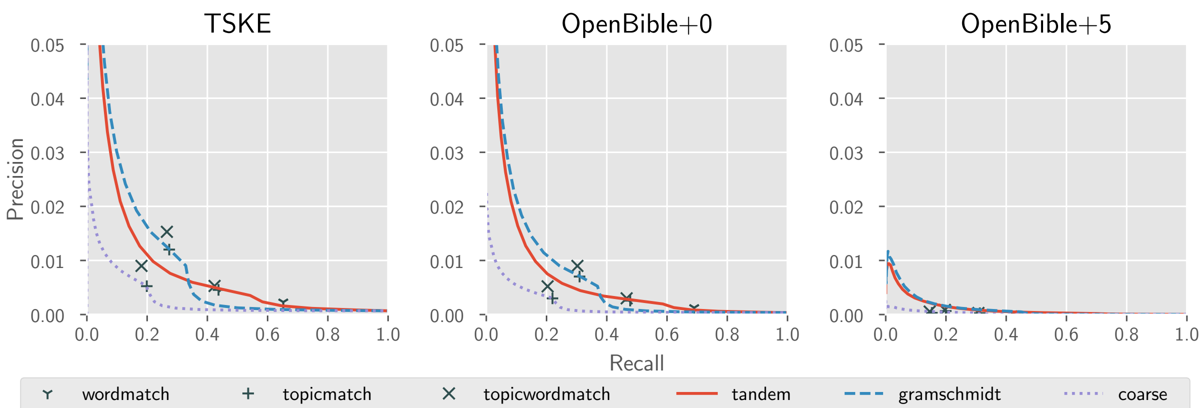


Figure 6.5: Cross-reference PRC plots with different models for cross-reference selection with three different datasets.

best for nearly all levels of false positive rate. However, the Gram-Schmidt based anchors produce slightly better true positive rates for very low false positive rates.

This trend is better illustrated with the PRC plots in Figure 6.5. While for higher values of recall the tandem anchor selection strategy does win out, it is only after precision significantly drops that tandem anchors produces superior predictions to Gram-Schmidt anchors.

6.3.3 Cost Analysis

While the precision-recall curves in Figure 6.5 may suggest that Gram-Schmidt based topic models produce superior topics for cross-referencing, we suggest that this analysis may be missing a key point in the real world analysis. As an alternative to both PRC and ROC curves, we suggest that this task might be best served with an analysis of cost per true positive.

We envision that our system would be used to produce a set of candidate cross-references which would then be curated using human annotators. These annotators would be tasked with evaluating each potential reference and determining whether or not each cross-reference is valid. Critically, the annotator would only be required to evaluate individual cross-references, not the entire text.

As a working example of the cost of such an annotation process, suppose we use a popular crowd-sourcing service (e.g., Amazon Mechanical Turk) to produce human annotations. We might reasonably expect to pay something around $0.01 USD per annotation. We would likely require some form of quality control in the form of redundant annotations, so we might end up paying $0.05 USD per annotated cross-reference candidate. Of course, the exact cost per annotated cross-reference will vary depending on the service and difficulty of the specific text being cross-referenced. However, we will use these estimates for the purpose of illustration.

Suppose as part of this working example, we are interested in producing a resource with 12,000 valid cross-reference annotations (matching the size of the previously mentioned LDS edition of the Bible [2]). Consulting Figure 6.6 we can then determine how many candidate cross-references we would need to produce for human annotation in order to create the final curated cross-reference resource. For example, using TSKE as our ground truth, we would need approximately 150,000 predicted positives in order to find 12,000 true positives. At $0.05 USD per annotation, this would cost about $7,500 USD. Supposing that our annotators were more selective, we could use the OpenBible+0 as the ground truth, which would roughly

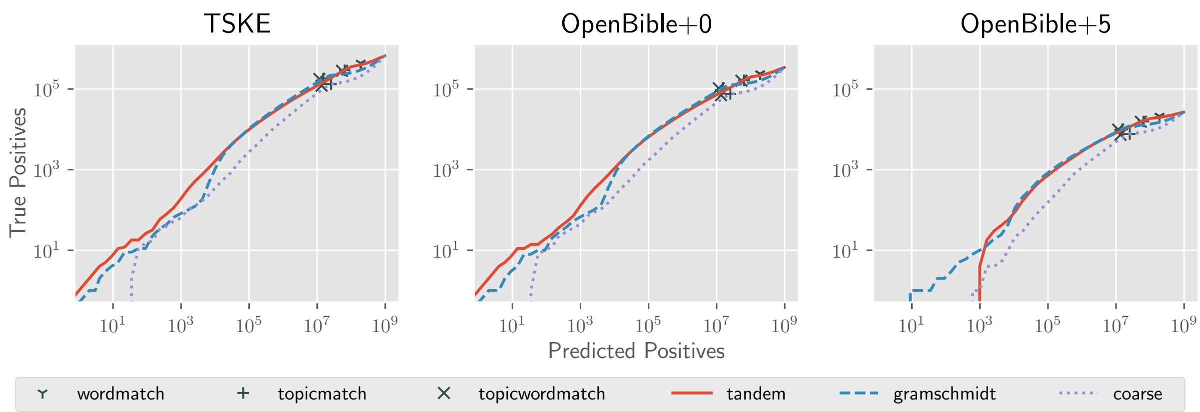


Figure 6.6: Cross-reference cost curves with different models for cross-reference selection with three different datasets. Note the log-log scale. The x-axis denotes the number of cross-references produced by our method, while the y-axis indicates how many of those cross-references are valid according to the human-provided ground truth.

double the cost. With OpenBible+5, which is considerably more selective, this cost rises to approximately $1,000,000 USD. In contrast, with traditional coarse-grained topic modeling, this cost is anywhere between $40,000 USD using TSKE as the ground truth, to $17,500,000 USD using OpenBible+5 as the ground truth.

While these costs may seem prohibitive, consider that the alternative is to have experts understand the entire text to the degree that they can read one passage and recall other relevant passages they have previously read. In the case of religious texts, this is often possible since adherents study those texts as part of their daily routine. For example, in the case of the LDS edition, it took a committee of experts seven years of work to produce their cross-reference resource, even with the aid of hundreds of volunteers. However, without those experts and volunteers, the cost would have been even greater. In the naive case where every possible reference is manually checked, the cost skyrockets to around $48,000,000 USD.

We also note that while Figure 6.6 shows that while Gram-Schmidt anchors does eventually win out against tandem anchors, for smaller numbers of predicted positives (less than 10^4), tandem anchors produces more true-positives. Depending on the needs of the user and the available annotation resources, this may be sufficient. Thus, for some smaller cross-referencing efforts, tandem anchors does provide some benefit.

evict wipe life death
reproach book dread shame
seek face find turn heart none found
said say lord servant heard brought behold israel
ransom sheol pit draw soul life death
pass away behold earth come wind midst day made king
shall people god bring day give land

Table 6.1: Topics which were attributed to tokens in Isaiah 25:8. See Figure 6.1 for the text of this verse and the most relevant cross-references for this verse. As per usual, each topic is represented as a set of related words, however rather than showing the top n most probable words in each topic, we show the words which had probability greater than $1e-4$ in the topic-word distribution, leading to unequal numbers of words shown for each topic.

6.3.4 Qualitative Example

Reusing the example of Isaiah 25:8 discussed in Section 6.2.1 and shown in Figure 6.1, we now examine the topics and cross references of a single verse to help us understand qualitatively how fine-grained topic modeling can help us to produce a cross-reference resource. While this example is not randomly selected, it was selected arbitrarily from a selection of well-known Bible cross-references for illustrative purposes. For this qualitative example, we use the cross-reference results gleaned from using tandem anchors, and cosine distance to predict the most topically similar verses. We focus on this single illustrative example, but note that other examples of verse pairs can be see in Appendix B.

Figure 6.1 shows the topics which were attributed to tokens of Isaiah 25:8. Unsurprisingly, we that see many of the important terms unique to this verse appear as the probable words in this topic. More importantly, we see that the topics reflect important cooccurrence patterns found across both this verse and some of the verses which are considered cross-references for Isaiah 25:8. For example, we have a topic which includes terms such as 'ransom', 'Sheol', and 'death', which are also terms which appear in the linked verse Hosea 13:14. Because of this shared topic between Isaiah 25:8 and Hosea 13:14, this cross-reference is ranked fairly highly, and would have been discovered after annotating about five hundred thousand predicted cross-references.

Pride and Prejudice, Ch. 8	**Emma, Ch. 4**
"I have an excessive regard for Miss Jane Bennet, she is really a very sweet girl, and I wish with all my heart she were well settled. But with such a father and mother, and such low connections, I am afraid there is no chance of it."	"I wish you may not get into a scrape, Harriet, whenever he does marry;–I mean, as to being acquainted with his wife–for though his sisters, from a superior education, are not to be altogether objected to, it does not follow that he might marry any body at all fit for you to notice. The misfortune of your birth ought to make you particularly careful as to your associates. There can be no doubt of your being a gentleman's daughter, and you must support your claim to that station by every thing within your own power, or there will be plenty of people who would take pleasure in degrading you."

Table 6.2: Two related passages identified by our cross-referencing system from the works of Jane Austen. Passage 1 is taken from the eighth chapter of *Pride and Prejudice*, and Passage 2 is taken from the fourth chapter of *Emma*. These two passages are a valid cross-reference because they both discuss social standing and family connections in the context of marriage. The connection was found even with their lack of shared words.

An important term which is missing from the topic-word distributions for Isaiah 25:8 is 'swallow'. As a consequence of this omission, the link between Isaiah 25:8 and 1 Corinthians 15:54, a significant cross-reference shown in Figure 6.1, is actually ranked fairly low, and would have been found only after evaluating more than 1.6 billion predicted cross-references. While these two verses did share one topic in common, most of the text of Isaiah 25:8 was attributed to topics not used with 1 Corinthians 15:54. This rare term (which as it turns out is rare enough that it has low probability in every topic), should have been the link between these two verses.

We suggest that future work might improve upon our work by boosting the importance of rare words, perhaps using something as simple as TF-IDF weights instead of raw word counts to compute word cooccurrences. Another possible avenue for improvement could involve intentionally selecting verses from which to form tandem anchors based on the presence of significant but rare terms.

6.4 Discussion

Without extensive cross-referencing resources for more secular datasets, it is difficult to empirically prove the usefulness of our system generally without an extremely costly user study. That said, we make a small attempt by examining cross-references generated from the complete works of Jane Austen. Based on our cost analysis in Section 6.3.3, and since we will be examining only a small number of cross-references, we utilize tandem anchors to generate topics.

We examine the first 300 cross-references produced by our system. Of these, we find that 39 of them are valid, linking passages from all six works by Jane Austen. As with our experiments with the Bible, this level of precision is sufficient that we believe that we could dramatically lower the cost of producing a full cross-referencing resource for this text.

In the first 300 cross-references, we note that 109 of these are cross-references linking a paragraph in the eighth chapter of *Pride and Prejudice* to other passages in our corpus. Of the references involving this one paragraph, 22 were valid. An example of such a cross-reference is shown in Table 6.2. While marriage in general is a common theme in the works of Jane Austen, this particular paragraph more specifically discusses the role of social status and family connection as it relates to choosing a marriage partner. We note that the connection between the passages in Table 6.2 is thematic; they share no significant words in common, demonstrating the capability of the system to detect nuanced topics and themes.

6.5 Conclusion

We have produced a system using fine-grained topic modeling which is able to propose candidate cross-references which can be verified by non-expert human annotators for the purpose of creating a cross-reference resource at a fraction of the cost of current manual techniques. Our method, which utilizes tandem anchors to produce large numbers of highly nuanced topics coupled with an effective assignment strategy, is able to produce document-topic vectors which are comparable using cosine distance.

Our results also demonstrate that this system would not be as cost effective with traditional coarse-grained topic modeling. While we can find sets of topically related documents using coarse-grained topics, for the task of finding the most closely related documents we require a system which is more specific. We suggest that this success serves as motivation for exploration of fine-grained topic modeling for other topic-based use cases which require nuance and precision.

Part IV

Parallelization

In this final part of the dissertation, we show that proper utilization of parallel hardware can significantly speedup anchor-based topic modeling. Of course, given sufficient time, this is not strictly necessary for obtaining our results, as all of our algorithms can be run in serial. However, as we increased the number of topics in our model, we found that the increased runtime of topic recovery did hamper our exploratory experimentation. Through parallelization of our algorithms, we were able to significantly reduce turn-around time between, reducing friction between invention and realization of our ideas. Our efforts in this space focused on Mrs, a lightweight MapReduce framework designed specifically for scientific computing. In Chapter 7, we briefly describe the MapReduce programs we employ during our experiments with the anchor words algorithm. In Chapter 8, we describe four modifications to the original MapReduce algorithm which make Mrs especially well-suited scientific computing.

Chapter 7

Parallelization of Anchor Algorithm

7.1 Introduction

Anchor-based topic models are a fast and scalable alternative to traditional probabilistic topic modeling. With the anchor algorithm, topic recovery takes $O(V^2K + K^2VT)$, where K is the number of topics, V is the size of the vocabulary, and T is the average number of iterations required for the inference procedure to converge [6]. In contrast, traditional probabilistic algorithms such as Latent Dirichlet Allocation typically require $O(MDKT)$ to run, where M is the number of documents and D is the expectation of the length of the documents [42]. This can be improved using online variational Bayes updates so that we require only a single pass through the data [47], but nevertheless, topic inference with LDA is expensive compared to the anchor words algorithm. Since K is typically small (< 100), and V typically grows logarithmically with respect to the size of the data [46], the anchor words algorithm scales to much larger datasets and is fast enough to facilitate interactive topic modeling [66].

Topic recovery with the anchor algorithm requires solving V quadratic programs of size $K \times K$, and each of those programs takes T iterations on average. Consequently, when K is large, as is the case in our experiments with fine-grained topic modeling, the runtime of the anchor words algorithm can become expensive.

In addition, topic recovery depends on a $V \times V$ cooccurrence matrix Q. We typically assume that this matrix is precomputed and fixed. However, computing Q takes $O(MD^2)$. For large datasets with millions of documents, this runtime can be considerable, even if topic recovery is fast once Q is constructed.

Happily, both topic recovery and the construction of the cooccurrence matrix Q are amenable to parallelization. In this chapter, we briefly describe our efforts in this space, and show that significant speedup is achievable for these algorithms.

7.2 Parallelization with Mrs

In our efforts to parallelize parts of the anchor algorithm, we utilize Mrs MapReduce, a lightweight Python-based MapReduce framework designed specifically for scientific computing. While the Mrs project is available online[1], we refer the reader to Dean and Ghemawat [30] for a high-level overview of the structure of MapReduce programs, and McNabb et al. [72] for an overview of the Mrs framework. Furthermore, we point to Chapter 8 for a description of our modifications to the MapReduce framework which make Mrs particularly well suited for scientific computing compared to other MapReduce implementations. In this chapter, we will only describe our algorithms at a high level, but we note that our code is available online for those interested in specific implementation details[2].

7.2.1 Q-construction

Following Arora et al. [6], the construction of the $V \times V$ cooccurrence matrix Q is computed as follows:

$$Q = \tilde{H}\tilde{H}^T - \hat{H} \tag{7.1}$$

where $\tilde{H}$ is a $V \times M$ sparse column normalized matrix encoding document-word counts, and $\hat{H}$ is a $V \times V$ diagonal matrix which corrects for words which cooccur with themselves. Assuming properly vectorized code, Q construction is fast and efficient.

However, for large datasets with millions of documents, $\tilde{H}$ may not fit in RAM, even with proper sparse representation. Leaving the details of the derivation to Liu et al. [62], as an alternative we can compute the contribution of each individual document towards the

[1] https://github.com/byu-aml-lab/mrs-mapreduce

[2] https://github.com/jefflund/ankura

Listing 7.1 MapReduce program for computing cooccurrence matrices in parallel.

```
def map(key, batch):
    Q_part = matrix(V, V)

    for document in batch:
        # following Equation 7.2, compute
        # contribution of each document in the batch
        Q_part += contribution(document)

    # yield partial Q, with empty since only one reduce task
    yield '', Q_part

def reduce(key, values):
    # sum the partial Q's
    Q = sum(values)
    # topic recovery uses row normalized Q
    yield row_normalized(Q)
```

final Q matrix. Given that $w_{d,i}$ is the ith word of document d, e_i is the ith basis vector in V-dimensional space, and n_d is the number of tokens in document d, then the contribution of each individual document is:

$$\frac{1}{Mn_d(n_d-1)} \sum_{i \neq j, i,j \in [n_d]} e_{w_{d,i}} e_{w_{d,j}} . \tag{7.2}$$

Using Equation 7.2, we can compute Q simply by summing the contribution of each individual document. Critically, this process can be done with just a single document in RAM at a time, meaning that while this computation is not as efficient, it scales to arbitrarily large datasets (assuming that the $V \times V$ matrix Q fits in RAM).

We note that this formulation is also trivially parallelized using MapReduce as shown in Listing 7.1. Note that each map task is actually responsible for multiple documents. The batch size for each map task is crucial for good performance. If the batch size is too small, the increased communication costs between the map tasks and the single reduce task will

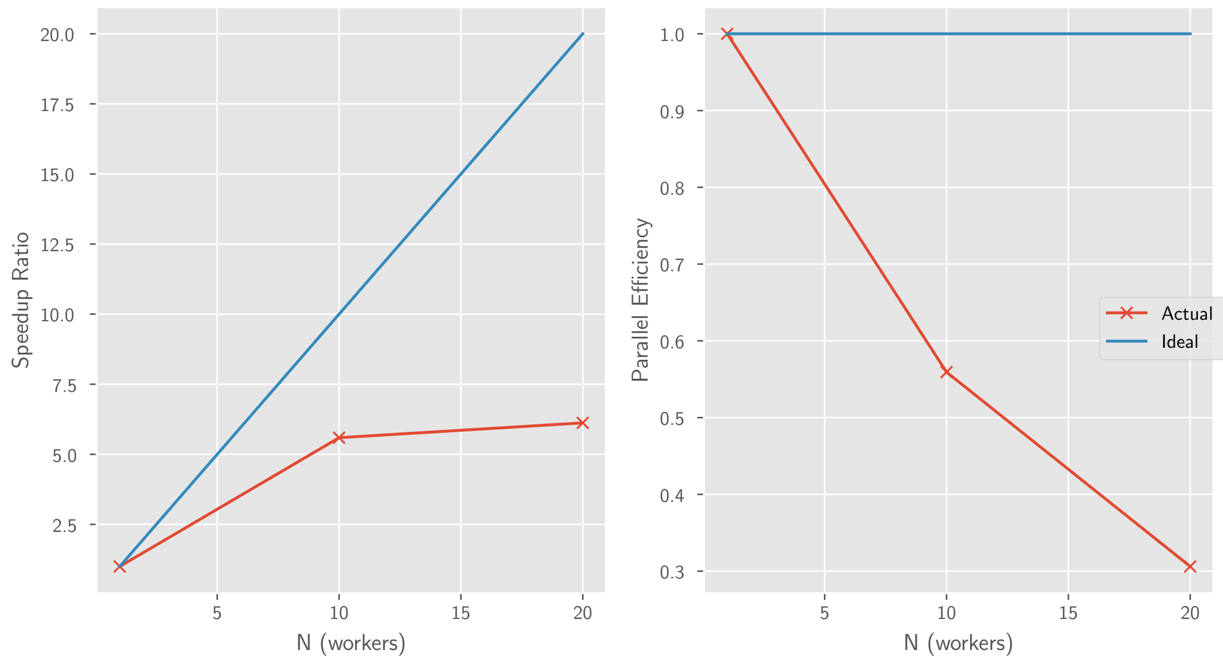


Figure 7.1: Speedup ratio and parallel efficiency for construction of cooccurrence matrix using Mrs MapReduce using one million Amazon reviews as the text.

hurt parallel efficiency. On the other hand, if the batch size is too large, then we cannot take as much advantage of the available parallel hardware.

We demonstrate the usefulness of this parallelization by constructing the cooccurrence matrix using the text of one million Amazon reviews[3]. We reduce the runtime of this computation from approximately 60 minutes to less than 10 minutes. However, as shown in Figure 7.1, parallel efficiency does drop significantly as we increase the number of workers. This is likely because in our experiments, the data is stored on a shared file system, and as we increase the number of workers, the increase in communication overhead lowers parallel efficiency.

[3]The reviews are sampled from the full set of reviews available at `http://jmcauley.ucsd.edu/data/amazon`.

7.2.2 Topic Recovery

With anchor-based topic recovery, each of the V words is represented as a row in $\bar{Q}$, which is a row normalized version of Q encoding conditional probabilities of observing one word given another. The V quadratic programs seek to represent each word as a convex combination of K anchors, which are also points in the same V-dimensional cooccurrence space. We represent these coefficients as a $V \times K$ matrix C. Under the anchor assumption, which states that each anchor uniquely identifies a topic, the coefficients in C are the conditional probability of a topic given a word. This is the inverse conditioning we desire, so we multiply this coefficient matrix by a vector giving the empirical probabilities of each word, which according to Bayes' rule yields the probability of a word given a topic.

Given a fixed Q, and a fixed set of anchors A, each of these quadratic programs is entirely independent of the others. Consequently, we can easily write a MapReduce program which performs topic recovery, as seen in Listing 7.2. Once again, some care must be taken with the size of the batches sent to each map task, since too few words per batch increases communication costs, while too many words per batch reduces parallel utilization.

We note that each map task output is sparse, and we can glean some savings in communication costs if we use an appropriate sparse matrix representation. We recommend that a sparse row format be used, since matrix addition is efficient in the computational bottleneck of the reduce task.

As seen in Figure 7.2, with our experiments in Chapter 6 with fine-grained topic modeling on the English Standard Version of the Bible, we are able to effectively leverage our moderate parallel resources. Using 20 workers, each using 8 cores of an Intel Core i7-4770K, we are able to bring the time required for topic recovery down to a median of 10 minutes. With a single worker, topic recovery takes a median of 128 minutes.

We note however, that these speedup numbers are only applicable for fine-grained topics. Recall that topic recovery in serial requires $O(V^2K + K^2VI)$, where I is the average number of iterations for gradient descent to converge when solving for rows of C. Because

Listing 7.2 MapReduce program for parallel topic recovery.

```
def map(key, batch):
    # Q and anchors are fixed, and are presumably accessible
    # on a shared filesystem.
    Q = get_Q()
    anchors = get_anchors()

    C_part := matrix(V, K)
    for word in batch:
        # Solve a quadratic program for each word in the batch,
        # representing each word as a combination of the anchors.
        C_part[word] = recover(Q[word], anchors)

    yield '', C_part

def reduce(key, values):
    # Compute emperical distribution over words
    Q = get_Q()
    prior = diag(Q.sum(axis=1))

    # Sum the partial C's
    C = sum(values)

    # Use Bayes's rule to complete topic recovery
    yield dot(prior, C)
```

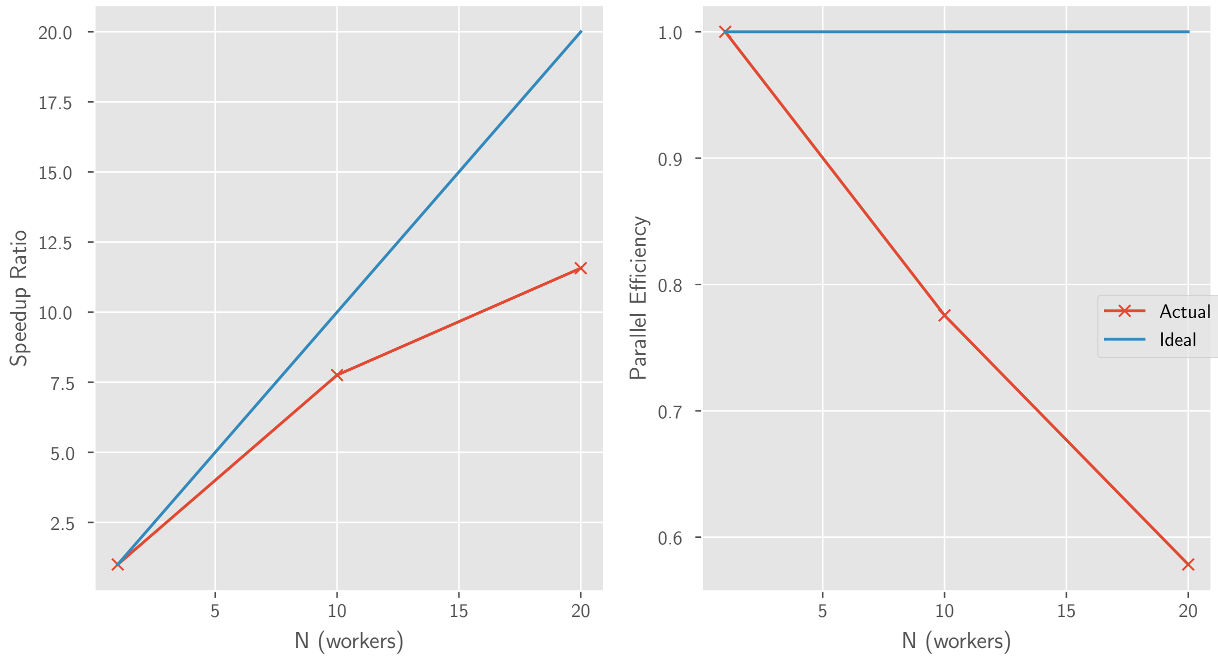


Figure 7.2: Speedup ratio and parallel efficiency for fine-grained topic recovery using Mrs MapReduce. We use 3,000 topics with Gram-Schmidt anchors on the English Standard Version of the Bible. See Chapter 6 for more details on the dataset and topics.

runtime scales quadratically with the number of topics K, there is more computational benefit from parallelization when K is large. In fact, when $K = 100$, we find that communication overhead becomes actually increases runtime using any more than 4 workers. Admittedly though, when K is small, runtime is already fast enough to be interactive [66], so parallelization is not actually needed in practice.

7.3 Conclusion

While the anchor words algorithm is certainly scalable enough to be used with a single processor, we have demonstrated that parallelization can significantly speed up the anchor words algorithm. The speedup for constructing cooccurrence matrices is significant on large datasets with millions of documents, although we note that this is a one-time computation which can be precomputed. However, the performance gains with topic recovery are significant from the standpoint of a researcher. The ability to quickly iterate through new ideas proved invaluable during the completion of this dissertation.

Chapter 8

Mrs: High Performance MapReduce for Iterative and Asynchronous Algorithms in Python

Published in Workshop on Python for High-Performance and Scientific Computing 2016 [64]

Abstract

Mrs [70] is a lightweight Python-based MapReduce implementation designed to make MapReduce programs easy to write and quick to run, particularly useful for research and academia. A common set of algorithms that would benefit from Mrs are iterative algorithms, like those frequently found in machine learning; however, iterative algorithms typically perform poorly in the MapReduce framework, meaning potentially poor performance in Mrs as well.

Therefore, we propose four modifications to the original Mrs with the intent to improve its ability to perform iterative algorithms. First, we used direct task-to-task communication for most iterations and only occasionally write to a distributed file system to preserve fault tolerance. Second, we combine the reduce and map tasks which span successive iterations to eliminate unnecessary communication and scheduling latency. Third, we propose a generator-callback programming model to allow for greater flexibility in the scheduling of tasks. Finally, some iterative algorithms are naturally expressed in terms of asynchronous message passing, so we propose a fully asynchronous variant of MapReduce.

We then demonstrate Mrs' enhanced performance in the context of two iterative applications: particle swarm optimization (PSO), and expectation maximization (EM).

8.1 Introduction

Mrs [70] is a previously published framework for MapReduce projects implemented in Python. It was shown to be easily accessible, easy to use, and readily available for a variety of environments, scheduling systems, and file systems. This ease of use and availability made it well suited for academic or research environments where it is common for users to have generic, private clusters available rather than dedicated MapReduce clusters. Regarding its performance, Mrs was shown to perform just as well or better than Hadoop for various problems, meaning the user need not sacrifice quality for simplicity.

However, iterative algorithms still suffered a significant performance penalty. Much of this penalty comes from overhead, such as communication time between nodes, writing to reliable storage, and delay between iterations. For algorithms with a single iteration, or those with very few iterations, this overhead is acceptable, but for larger iterative algorithms, it becomes excessive. For example, given a large data set, a one second overhead per iteration may be insignificant for an algorithm requiring only one iteration, but one second per iteration for thousands of iterations on the same set could increase the execution time of a CPU-bound algorithm by hours.

We propose a set of modifications to the original Mrs framework in order to improve its performance on iterative algorithms. The first is to use direct communication between nodes for most iterations, and only occasionally write to reliable memory (Section 8.3.1). Second, we propose that reduce tasks be agglomerated with the subsequent map tasks with the same key, which reduces communication and halves the number of tasks that must be assigned each iteration (Section 8.3.2). Third, we present a generator-callback model for submitting operations for concurrent and asynchronous evaluation (Section 8.3.3). This model makes it easy for iterative algorithms to submit intermittent operations, such as convergence checks, to be evaluated concurrently without requiring significant bookkeeping. Finally, we introduce an asynchronous extension of the MapReduce programming model in Section 8.4 which efficiently supports algorithms such as Particle Swarm Optimization (PSO) [71] where iteration can proceed at a different rate for each key. This model allows the same straightforward map and reduce functions to work in both synchronous and asynchronous operation.

While we consider the generator-callback function novel to our design, the rest of these modifications have already seen success in other publications. We discuss these in Section 8.2.

Finally, we demonstrate the application of these techniques in Mrs [70]. Section 8.5 evaluates the performance of Mrs with and without these features, using PSO [71] and expectation maximization (EM) [25] as examples, and shows significant improvements in performance. Compared to standard MapReduce, using a reduce-map operation improved PSO performance by 31%. For EM, iterations without checkpointing to redundant storage show a 91% improvement, making parallelization feasible, and the reduce-map operation gives an extra 11%. Asynchronous MapReduce improves performance of PSO by an additional 24% in the presence of moderate variability in task execution times for a total gain of 53%. Furthermore, it performs iterations faster than synchronous PSO even when task execution times are uniform. With 768 processors and uniform tasks, Asynchronous MapReduce increases the throughput by 47%.

8.2 Related Work

MapReduce [30] is a popular framework for performing parallel processes, with Hadoop being its most well known and widely used open source implementation. Due to its limitations on iterative algorithms, however, several attempts have been made to modify MapReduce, or come up with a novel parallel processing framework, for the purpose of accommodating them. Most improvements or modifications consist of either modifying the the programming model, reducing communication, or optimizing the task scheduler.

MapReduce is technically defined as a map phase followed by a reduce phase, and this model must be extended, at least trivially, to support iterative programs. In most MapReduce systems, a "user program" or "driver" submits a job consisting of a map phase and a reduce phase, waits for it to complete, reads the results, and then repeats. Several

MapReduce-like systems allow the user to specify an arbitrary directed acyclic graph of data dependencies [21, 53, 82, 120]. Frameworks like Maiter [123] and GraphLab [63] have implemented novel models specifically directed at iterative parallel processing. Maiter uses a directed acyclic graph to represent data and dependencies, but instead of updating the data at each iteration, it only keeps track of the changes in the data from iteration to iteration. This method of iterating makes asynchronous task scheduling simple and eliminates wasteful processing. GraphLab represents a given problem with its own type of directed graph along with a shared data table to represent information common to multiple tasks.

Reducing communication has been a common modification to MapReduce because of the amount of excess overhead that is generated with iterative algorithms. One strategy for this is to store data locally. Conch [124] and Twister [36] do this. Conch stores all data in local cache and uses a memory manager to optimize total memory use. It only writes to an HDFS when memory overflows or when the algorithm terminates. Twister pushes intermediate data directly from map tasks to reduce tasks and stores data on the master between reduce and map tasks. Many other frameworks take advantage of this concept in some way [20, 79, 101]. Both Conch and Twister also combine certain tasks, sending data directly from one task to another instead of having each task read from memory, compute, and then write back to memory like in normal MapReduce.

Intelligent task scheduling can also help improve performance. Several frameworks have developed optimized task schedulers that take advantage of specific modifications in their framework or plan ahead to reduce waiting and communication [20, 79, 94, 124]. Another technique implemented in iMapReduce [121, 122] aims to eliminate most of the overhead by making all tasks persistent. It seems, however, that the ideal scheduling would be a form of asynchronous scheduling. iHadoop's [37] primary modification to Hadoop was the addition of asynchronous scheduling, but several frameworks have since implemented some sort of asynchronicity into their designs [101, 121–123].

We build on these concepts and apply our modifications to Mrs, resulting in a convenient, high-performance Python implementation of an iterative MapReduce framework. Each of the previously mentioned concepts is implemented in Mrs using methods described Sections 8.3 and 8.4, as well as the generator-callback model to handle task scheduling and completion.

8.3 Synchronous MapReduce

Iterative programs are sensitive to overhead such as communication costs because such overhead accumulates from iteration to iteration. We propose three improvements to reduce overhead. Section 8.3.1 shows a principled approach for limiting the frequency of checkpoints to distributed storage. Section 8.3.2 describes a reduce-map operation for agglomerating reduce and map tasks. Section 8.3.3 defines a generator-callback model for defining a directed acyclic graph of operations in an iterative program. These three improvements are evaluated later in the paper in Section 8.5.

8.3.1 Infrequent Checkpointing to Distributed Filesystems

Traditional MapReduce implementations communicate all intermediate data through a distributed filesystem. Such filesystems replicate all data to ensure fault tolerance but come with a significant performance penalty. Communication and storage in MapReduce should explicitly address the tradeoff of speed vs. capacity and fault tolerance. An ideal runtime would be able to automatically move data between levels of the memory hierarchy, a well-known strategy for storage devices [**?**]. While an advanced automatic memory hierarchy may be impractically complex for a MapReduce system, communicating data from some iterations directly between nodes and storing data from other iterations to reliable storage is a simple way to balance speed and fault tolerance.

We advocate storing the output of most map and reduce tasks on the local filesystem, while storing the output from occasional checkpoint iterations to reliable storage. The operating system buffers data on the filesystem in RAM and automatically migrates it to disk if necessary. With fast iterations, short-lived intermediate data is usually deleted before ever being written to disk. This approach provides the speed of RAM when possible and gracefully sacrifices speed for capacity when the size of data is great. In the event that a node fails and makes its local storage unavailable, a MapReduce runtime can roll back to the most recent checkpoint iteration.

In almost any realistic iterative program, checkpointing should occur far less than every iteration, unlike most MapReduce systems, including Hadoop. Some other implementations, like Twister [36], go to the other extreme and do not support distributed storage, sacrificing fault tolerance. The ideal checkpointing frequency depends on the expected cost of failures vs. the cost of redundancy. We estimate and compare these costs using a simple model. While specific circumstances may warrant a customized model to determine the ideal checkpoint frequency, this simple model gives a rule of thumb and demonstrates the cost of checkpointing every iteration.

In this simple model, failures are assumed to be independent. We also assume that the times required to compute an iteration, perform a checkpoint, or initiate a recovery are constant. Let n be the number of iterations between checkpoints, t the time to perform each iteration, c the extra time required for a checkpointed iteration, and r the time to initiate recovery after a failure. Let X be a Bernoulli-distributed random variable indicating whether a failure occurs during an iteration, with probability determined by the product of the mean time between failures in a cluster f and the total time per iteration (including the amortized cost of checkpointing):

$$X \sim Bernoulli\left(\frac{1}{f}\left(t+\frac{c}{n}\right)\right)$$

Let $Y \sim Uniform(n)$ be a random variable indicating the number of iterations since the last checkpoint, which is independent of X. Then the expected value of the number of seconds of extra work in an iteration is:

$$E\left[X\left(r+Yt\right)\right]=\frac{1}{f}\left(t+\frac{c}{n}\right)\left(r+\frac{n}{2}t\right)$$

If this is less than the amortized cost of checkpointing per iteration ($\frac{c}{n}$), then redundancy costs more than it helps. The breakeven point is given by solving for n:

$$n=\max\left[1,\frac{1}{t}\left(\sqrt{\left(\frac{c}{2}+r\right)^2-2c(r-f)}-\left(\frac{c}{2}+r\right)\right)\right]$$

Most reasonable values cause n to be larger than 1. For example, suppose that writing to reliable storage adds 10 seconds per iteration ($c = 10$) and that initiating recovery from a checkpoint requires 60 seconds ($r = 60$). Note that the values for c and r are conservative, and increasing c or decreasing r would increase n. For a program with moderately slow one-minute iterations ($t = 60$) and frequent failures on average once every three hours ($f = 10800$), the breakeven point n is 6.7. For a program with fast iterations ($t = 1$) and a moderate failure rate of one failure in a cluster per week ($f = 604800$), the breakeven point n rises to 3413. The actual ideal frequency of checkpointing depends on individual circumstances, and many short-running programs may not require checkpointing at all.

8.3.2 Reduce-map Operation

Iterative MapReduce programs consist of a string of iterations, each with a map operation and a reduce operation. The new task dependencies between iterations motivate rethinking the decomposition of work into tasks. The output from each reduce task is the sole input to a single map task in the next iteration. Some systems take advantage of this relationship between tasks by scheduling them to the same processor or starting a map task before all preceding reduce tasks are complete [37, 121]. We instead agglomerate each reduce task with the map task that uses its output, which removes this communication and halves the number of tasks that the master must assign each iteration. Figure 8.1 shows the dependencies between

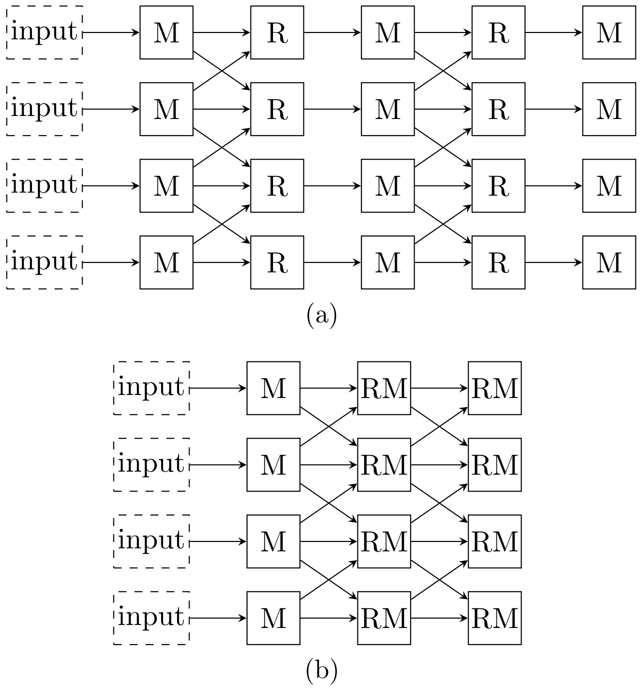
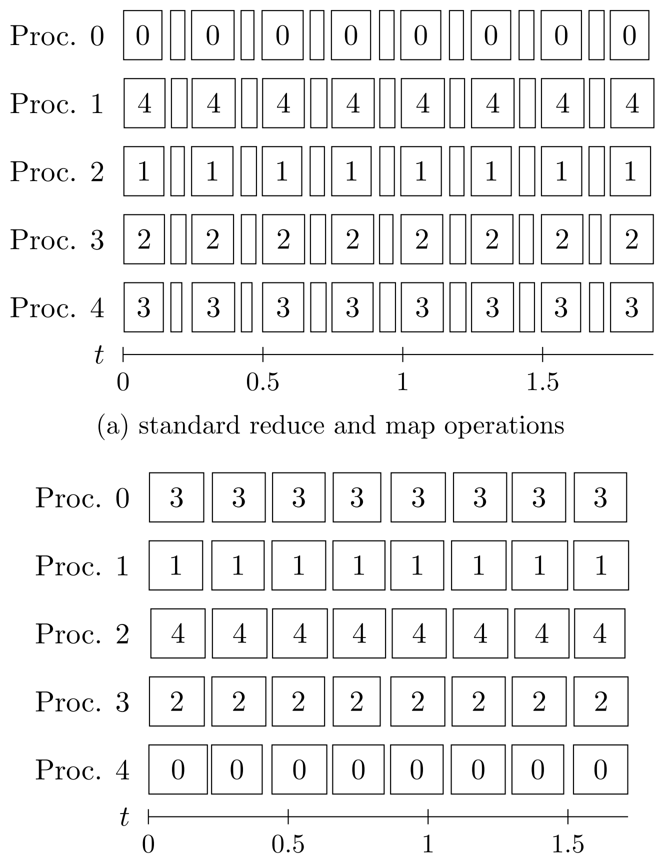


Figure 8.1: Task dependencies of a typical iterative MapReduce program with (a) standard map (M) and reduce (R) operations, contrasted with (b) combined reduce-map (RM) operations.

tasks with separate reduce and map tasks (Figure 8.1a) and with combined reduce-map tasks (Figure 8.1b).

In principle, the master might be able to autodetect these fine-grained data dependencies, but we allow the user to either specify a reduce-map dataset or separate reduce and map datasets. The user still provides a map function and a reduce function, but specifying a reduce-map operation allows the runtime to combine tasks and eliminate communication.

Figure 8.2 demonstrates the difference between MapReduce using separate reduce and map operations and MapReduce using combined reduce-map operations. Combining the reduce and map eliminates the time spent in assigning each reduce task and waiting for it to complete.



(a) standard reduce and map operations

(b) combined reduce-map operations

Figure 8.2: Actual task execution traces generated from a sample application (particle swarm optimization, see Section 8.5.1 for details) without and with combined reduce-map tasks. The run with reduce-map operations avoids the overhead of an independent reduce task and completes sooner. The horizontal axis is measured in seconds, with the left and right sides of each box aligning with the task's start and stop times. The number in each box is the key of the map task.

8.3.3 Iterative Programming Model

The standard MapReduce model defines a single map phase followed by a single reduce phase [30], but iterative programs execute an arbitrary number of operations and often need to compute a loop termination condition that depends on the results. Computing convergence checks infrequently and concurrently with subsequent iterations improves performance, but most MapReduce implementations do not provide any mechanism to specify this behavior. We propose an alternative model for defining operations that allows programs to specify complex behavior without becoming inherently complicated. This model is available for iterative programs that require such behavior but is not required for traditional single-iteration MapReduce programs.

Varying the operations that are performed each iteration—for example, only performing convergence checks or printing intermediate output occasionally—can significantly improve performance. Suppose a program runs one second per iteration and that evaluating the loop termination condition requires a tenth of a second. If this loop condition computation is performed every iteration, it adds about 6 minutes over the course of an hour. Reducing the check to once per minute extends execution by an average of 30 iterations but still saves about 5 minutes total.

We represent parallel computation with a directed acyclic graph of datasets. A *dataset* represents data to be produced along with the associated operations required to produce it. In the representation of computation as a directed acyclic graph, the edges are the work, and the vertices are the data. Such datasets are similar in spirit to resilient distributed datasets [120]. When a user program submits datasets for asynchronous evaluation, the runtime performs computations in any order consistent with the dependency graph. Unlike the lazily evaluated tasks in Ciel [82], these datasets are evaluated eagerly. Because the next iteration can begin before evaluation of the loop condition completes, both operations can be performed concurrently. Likewise, the runtime can begin work on subsequent iterations while a user program is collecting and printing intermediate results. Unfortunately, manually managing a backlog of submitted datasets is tedious and error-prone, particularly if the work varies between iterations.

We propose a generator-callback model for submitting an arbitrary directed acyclic graph of asynchronously evaluated datasets and for handling their completion. The generator-callback model requires the program to provide a `generator` method. The `generator` method serves as an iterator or coroutine that produces work to be done. It submits each dataset for computation, along with an optional callback function to be called when computation completes. The master keeps a backlog of pending datasets, and if the backlog gets full, the generator blocks when it submits a dataset, later resuming when the backlog shrinks. Implementation is especially straightforward in languages that natively support coroutines, such as Python. As each dataset completes, the master calls the associated callback method, which can optionally read and process the results in parallel with subsequent MapReduce iterations. Termination is triggered either by the backlog exhausting after the generator completes or by a callback function returning `False` to indicate that the loop termination condition has been met. This model allows the MapReduce system itself to manage the backlog of datasets rather than exposing the details to the user. Manually maintaining a backlog requires bookkeeping that runs contrary to the simplicity of MapReduce.

Programs using the generator-callback model have greater flexibility and performance. This model is optional but may provide significant benefits for iterative programs that use it. Listing 8.1 is a program which submits one MapReduce step at a time. Unfortunately, the structure of this program forces computation to wait while the master blocks on pending operations, performs the convergence check, and outputs intermediate results. Listing 8.2 uses a generator-callback API to gain flexibility and performance. Note that in this example, an operation is submitted in the form of a declaration of the dataset it is to produce, not the operation itself. The generator function submits several iterations in advance, pausing only when the submit call (or yield statement) blocks. This allows tasks to be assigned with lower latency. The generator function also runs convergence checks with limited frequency

Listing 8.1 The structure of a generic iterative program using a standard iterative-unaware MapReduce API.

```
run_batches():
    # Intialize key value pairs with empty data.
    init_file = makeTempPath()
    for element_id = 1 to NUM_ELEMENTS
        init_file.writePair(element_id, "")

    # Perform mapreduce to obtain initial data.
    job = new_job()
    job.setInput(init_file)
    job.setMapper(init_map_func)
    job.setReducer(identity_reduce_func)
    data_path = makeTempPath()
    job.setOutput(data_path)
    job.waitForCompletion()
    last_data = data_path

    # Perform mapreduce iteratively.
    for iteration = 1 to MAX_ITERATIONS
        # Run a mapreduce iteration and wait for a dataset.
        job = new_job()
        job.setInput(last_data)
        job.setMapper(map_func)
        job.setReducer(reduce_func)
        data_path = makeTempPath()
        job.setOutput(data_path)
        job.waitForCompletion()
        last_data = data_path

        # Occasionally output and run convergence check.
        if iteration % CHECK_FREQUENCY = 0
            # Iteration stalls until this completes in serial.
            data = readAllFiles(data_path)
            perform_output(data)
            if converged(data)
                break
```

Listing 8.2 The structure of a generic iterative program using a generator-callback MapReduce API for performance and flexibility.

```
generator(queue):
    # Intialize key value pairs with empty data.
    kv_pairs = empty list
    for element_id = 1 to NUM_ELEMENTS
        kv_pairs.append(element_id, "")

    # Submit request to initialize curr_data.
    curr_data = MapDataset(kv_pairs, init_map_func)
    queue.submit(curr_data, NULL)

    for iteration = 1 to MAX_ITERATIONS
        # Submit asynchronous request to map iterm_data.
        interm_data = MapDataset(curr_data, map_func)
        queue.submit(interm_data, NULL)

        # Submit asynchronous request to reduce curr_data.
        curr_data = ReduceDataset(interm_data,
                                  reduce_func)

        # Occasionally submit output or convergence check.
        if iteration % CHECK_FREQUENCY = 0
            # Iterations continue in parallel with callback.
            queue.submit(curr_data, output_callback)
        else
            queue.submit(curr_data, NULL)

output_callback(data):
    data.readAllFiles()
    perform_output(data)

    # Continue processing if not converged.
    return !converged(data)
```

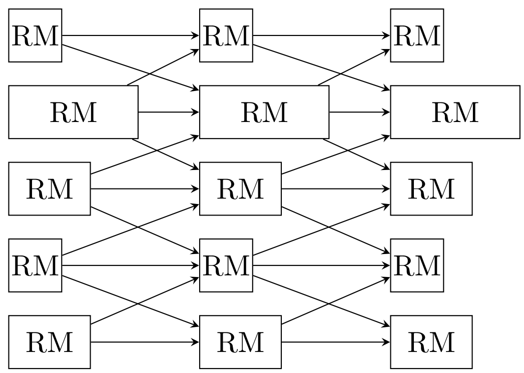
to reduce overhead. These convergence checks are performed concurrently with subsequent iterations and could be submitted as datasets if they represent significant computation. The simple generator-callback structure makes it easy to specify computation that varies from iteration to iteration and to read data asynchronously as computation completes. Both the blocking program and the generator-callback program use the same simple map and reduce functions.

8.4 Asynchronous MapReduce Programming Model

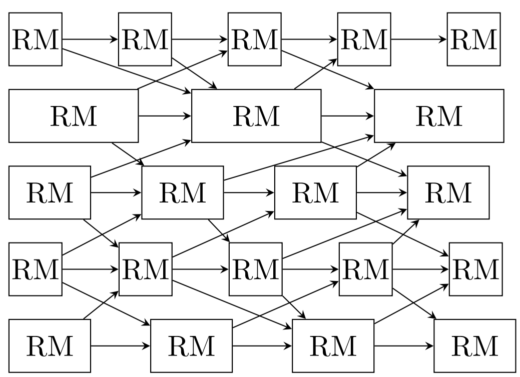
Iterative MapReduce can serve as a simple message passing framework. A map task serves to update an object, emit it, and emit messages to other objects. Between map tasks and reduce tasks is an implicit barrier for communication to complete, and a reduce task aggregates messages and emits the object, updated with information from the messages. With a reduce-map operation, the second implicit barrier, between the reduce and the following map, is removed. In the context of message passing algorithms, the MapReduce framework conceptually manages all communication, leaving map and reduce functions focused on the essence of the algorithm.

Not all iterative message passing algorithms require a barrier between each map operation and the following reduce. Such algorithms take advantage of all of the messages that have been received so far, and consideration of late-arriving messages is delayed to the next iteration. This class of algorithms is not expressible in the standard MapReduce programming model. Figure 8.3 illustrates task dependencies in an iterative program with heterogeneous task execution times.

In synchronous MapReduce (Figure 8.3a), the barrier betweeen iterations leaves the faster processors idle, but in asynchronous MapReduce (Figure 8.3b), the faster processors evaluate more iterations. The benefit can be similar on homogeneous processors if the map and reduce execution times vary or if there are a large number of processors.



(a) synchronous



(b) asynchronous

Figure 8.3: Task dependencies for reduce-map tasks in synchronous and asynchronous iterative MapReduce. Asynchronous MapReduce makes much more efficient use of processors.

We extend the MapReduce programming model to allow asynchronous message passing algorithms. In Asynchronous MapReduce, the programmer may specify that computation of a dataset may begin before all of the tasks in its parent have completed. Unfinished tasks continue execution, and upon completion, their results are added to a subsequent dataset specified by the programmer. The runtime framework keeps track of messages sent to keys with uncompleted tasks and ensures that they do not get lost. These pending messages are included in the same dataset as the results of the task when it eventually finishes. This simple model assumes only that a key refers to a specific object that remains fixed in each iteration. It works for programs that require multiple map and reduce phases in each iteration, and it is compatible with optimizations like the reduce-map operation.

Adapting a message passing MapReduce program to the asynchronous model requires the programmer to be aware of three new parameters to datasets:

- `async_start`
- `blocking_ratio`,
- `backlink`.

The `async_start` parameter is a boolean indicating whether a dataset can start asynchronously while some tasks in its input are still running. The `blocking_ratio` parameter determines the minimum fraction of tasks that must be completed before any child dataset can start asynchronously and defaults to 1 (fully synchronous). The `backlink` parameter specifies an earlier dataset from which uncompleted tasks are inherited. New tasks are only started for those keys whose corresponding tasks in the `backlink` dataset were completed before any asynchronous execution of its children began.

Although implementation of this model in the runtime framework is not quite trivial, its effect on the map and reduce functions is minimal. The semantics of the map function is unchanged. It still updates an object, emits it, and emits messages. The reduce function, however, is no longer guaranteed to be given the object at every iteration. It might receive only messages intended for the object. In iterations where the reduce function does not receive the object, it can combine messages together, but these messages cannot be incorporated into the object yet. Note that a program that works with Asynchronous MapReduce can also run in traditional synchronous mode.

Particle swarm optimization (PSO), described in more detail in Section 8.5.1, is an example of a simple iterative message passing algorithm that is naturally expressed in MapReduce [71]. The map function updates the position of a particle, emits the updated particle, and emits messages to neighboring particles. The reduce function aggregates the messages from neighboring particles, and emits the particle with updated information about its neighbors. Asynchronous parallel PSO is a variant of PSO which allows the evaluation of a particle to proceed even if messages have not been received from all of its neighbors [55, 109]. The fully distributed variant of asynchronous parallel PSO makes its message passing nature particularly clear [100].

Adapting a MapReduce implementation of parallel PSO to the asynchronous model requires very few changes. The reduce function must be tolerant of input that includes several messages but no complete particle; in this case it simply emits the best message. Assuming that this case is correctly handled, the map and reduce functions are identical to those in the synchronous MapReduce PSO implementation. The driver must be updated only to include the asynchronous MapReduce parameters. The map dataset at each iteration must be specified with a `blocking_ratio` below 1 and with a `backlink` pointing at the map dataset from the previous iteration. The reduce dataset at each iteration must be specified with the `async_start` parameter set to true. In the case that a single reduce-map dataset is used, it must be given all of these options.

Figure 8.4 shows the improved efficiency of Asynchronous MapReduce compared to synchronous MapReduce for tasks with variable execution times. In synchronous MapReduce, all tasks in an iteration start at the same time, which is limited by the end time of the slowest task in the previous iteration. In Asynchronous MapReduce, each task can start as soon

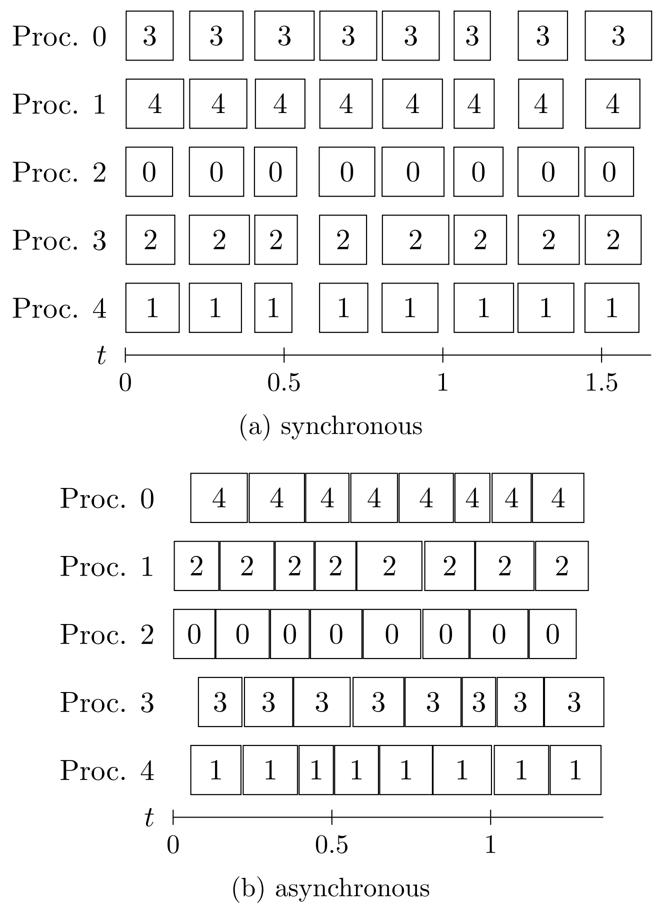


Figure 8.4: Actual task execution traces for PSO with synchronous and Asynchronous MapReduce. The horizontal axis is measured in seconds.

as the corresponding task from the previous iteration completes. Also note that the time between tasks is slightly less in asynchronous MapReduce, presumably due to the load on the master and the traffic on the network being less bursty.

8.5 Experimental Results

Although the approaches described in this paper are applicable to any MapReduce implementation, we evaluate their effects using the Mrs [70] framework. Experiments are performed on two clusters: a 2560-core cluster of 320 nodes, each with two quad-core 2.8 GHz Intel Nehalem processors and 24 GB of memory, and a 150-core cluster of 25 nodes, each with a 6-core 3.2 GHz AMD Phenom II X6 1090T processor with 16 GB of RAM, We run Mrs with and without various techniques enabled, compare the average time per iteration, and measure the average parallel efficiency per iteration. Parallel efficiency is the speedup per processor, relative to the fastest serial algorithm [40], for which we use typical serial implementations.

8.5.1 Synchronous MapReduce

In addition to the serial baseline, we compare with a baseline parallel configuration. This configuration uses redundant storage and convergence checks in serial every iteration, as is common in most MapReduce frameworks, but it also performs some optimizations, such as locality-aware scheduling, which are unavailable in some frameworks.

Although most users will wish to use redundant storage and perform convergence checks, these do not need to be run every iteration. Even if the occasional iteration cannot take advantage of the improved performance, the majority of iterations are accelerated. Section 8.5.1 describes particle swarm optimization (PSO) and shows the parallel efficiency of parallel PSO in MapReduce with the cumulative effects of direct communication, concurrent convergence checks, disabled convergence checks, and combined reduce-map tasks. Section 8.5.1 describes the EM algorithm and shows similar cumulative improvements.

Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an empirical function optimization algorithm inspired by simulations of flocking behaviors in birds and insects [19, 54]. The algorithm simulates the motion of a set of interacting particles within a multidimensional space. At each iteration, a particle moves and evaluates the objective function at its new position. A particle is drawn toward the best value it has seen and the best value that any of its neighbors has seen. PSO can be naturally expressed as a MapReduce program, with the map function performing motion simulation and evaluation of the objective function and the reduce function calculating the neighborhood best by combining the updated particle with messages from its neighbors [71]. For computationally inexpensive objective functions, task granularity is too fine if each map task operates on a single particle. In this case, a swarm can be divided into several subswarms or islands, and each map task operates on several iterations of a subswarm of particles [93, 99].

Listing 8.3 is an implementation of PSO using a generator-callback API as in Listing 8.2 from Section 8.3.3.

We find significant performance improvements for PSO in MapReduce. We use PSO with subswarms of 5 particles applied to the 250 dimensional Rosenbrock function [105]. Each subswarm runs for 50 “subiterations” in each map task. A baseline serial implementation of PSO takes an average of 0.26 seconds to simulate 5 particles for 50 iterations. Note that unlike the parallel implementation, this serial baseline does not serialize the state of particles between iterations. Combining reduce and map operations into a single reduce-map operation significantly reduces the overhead of assigning tasks. With separate reduce and map operations, the average time per iteration is 0.79 seconds. With a combined reduce-map operation, the average time per iteration drops to 0.55 seconds. This represents a reduction of 30.7% in each iteration.

Even a most inefficient MapReduce implementation would be able to provide reasonable parallel efficiency for a large enough problem size, but features that take into account the

Listing 8.3 PSO program using a generator-callback MapReduce API.

```
def run(self, job):
    job.default_reduce_tasks = NUM_PARTICLES
    job.default_reduce_splits = NUM_PARTICLES
    self.check_datasets = set()
    IterativeMR.run(self, job)

def producer(self, job, iteration):
    if iteration == 0:
        kvpairs = []
        for i in range(NUM_PARTICLES):
            kvpairs.append(i, '')
        start_data = job.local_data(kvpairs)
        self.swarm_data = job.map_data(start_data,
                self.init_map)
        start_data.close()
    elif iteration <= MAX_ITERS:
        tmp_data = job.map_data(self.swarm_data,
                self.pso_map)
        self.swarm_data.close()
        self.swarm_data = job.reduce_data(tmp_data,
                self.pso_reduce)
        tmp_data.close()
        if iteration % CHECK_FREQ == 0:
            tmp_data = job.map_data(self.swarm_data,
                    self.collapse_map, splits=1)
            check_data = job.reduce_data(tmp_data,
                    self.findbest_reduce, splits=1)
            self.check_datasets.add(check_data)
    else:
        return []

def consumer(self, dataset):
    if dataset in self.check_datasets:
        self.check_datasets.remove(dataset)
        dataset.fetchall()
        self.output(dataset.data())
        if self.converged(dataset.data()):
            return False
    return True
```

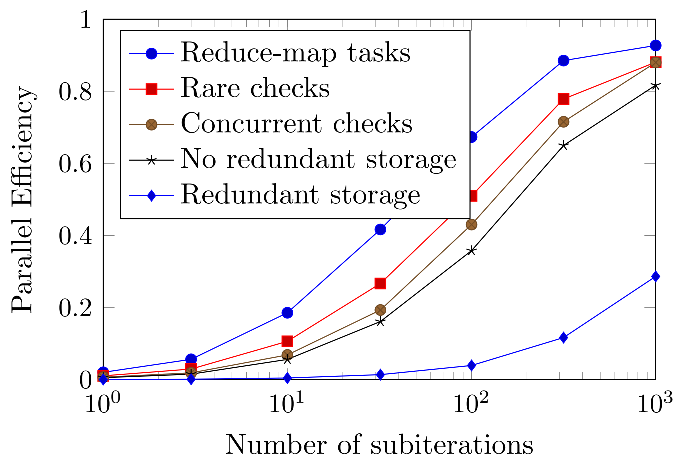


Figure 8.5: Parallel Efficiency (per iteration) of PSO in MapReduce with a sequence of cumulative optimizations. The x-axis represents the problem size (the number of subiterations in each map task). "Redundant storage" represents the baseline performance, with all data stored to a redundant filesystem and with convergence checks occurring after each iteration. "No redundant storage" shows performance for iterations with data communicated directly between processors. "Concurrent checks" shows further improvements when the convergence check is performed alongside the following iteration's work. "Rare checks" avoids unnecessarily frequent convergence checks. Finally, "reduce-map tasks" agglomerates each pair of reduce and map tasks into a single reduce-map task.

nature of iterative algorithms are able to extend the range of reasonable performance to more modestly sized problems. Figure 8.5 demonstrates the benefits of several techniques with respect to the problem size, which in the case of PSO is the number of subiterations performed by each subswarm within each map task. Note that the improvements are cumulative and optional. Though the figure only shows the performance of a reduce-map task in conjunction with direct communication, a configuration using redundant storage would still benefit from using combined reduce-map tasks. Furthermore, a program need not be equally efficient in each iteration. For example, even if redundant storage and convergence checks are performed occasionally, the majority of iterations can benefit from these optimizations. In MapReduce implementations that make redundant storage optional, a program only pays for the level of redundancy it needs.

Expectation Maximization

Expectation Maximization (EM) is an iterative algorithm commonly used to optimize parameters of finite mixture models in order to maximize the likelihood of the observed data [32]. Specifically, we apply the algorithm to a mixture of multinomials model in the context of clustering text documents [74, 111]. For each multinomial component in the model, we must maintain vectors with the same dimensionality as the number of features, which can be large. This greatly increases the communication cost when running in parallel, making efficiency difficult to obtain. Other mixture models, such as mixture of Gaussians, have much smaller parameter sizes, and have been parallelized successfully with the EM algorithm [57, 69]. We choose this particular model because it is inherently difficult to parallelize. With redundant storage and convergence checks at every iteration, performance was abysmal. However, the suggested improvements give much better parallel efficiency.

A single iteration of the EM algorithm consists of two steps. For our model, the expectation step (E-step) uses the current state of the parameters to estimate partial label assignments for the data. This is followed by the maximization step (M-step), which re-estimates the parameters using those partial label assignments. This algorithm is guaranteed to never decrease the log-likelihood of the data and will always converge to a local maximum.

EM for mixture of multinomials can be expressed as a two-stage iterative MapReduce program. The first stage of the program performs the E-step. Each map processes a shard of the documents and computes a posterior distribution given the current state parameters. The reduce then combines the posterior into partial counts for each of the labels. The second stage of the program re-estimates the parameters of the model. The map task performs normalization for each of the labels, and then the reduce task combines the normalized counts to produce the updated model parameters.

We tested the MapReduce implementation of EM with the 20 newsgroups dataset, a common benchmark for document clustering [58]. After preprocessing, the dataset had a vocabulary size of approximately 80,000 unique words. As a final step, we applied random

Table 8.1: Parallel efficiency per iteration of EM for various feature set sizes. As expected, higher feature set sizes lead to lower parallel efficiency, but removing redundant storage significantly helps. Further gains are realized by reducing convergence checks and using the reduce-map operation.

Optimization	80	252	8000	25298
Reduce-map tasks	0.411	0.357	0.277	0.193
Rare checks	0.362	0.314	0.253	0.18
Redundant storage	0.013	0.013	0.013	0.012

feature hashing, which maps each unique word to a predefined number of bins. Although simple, this type of feature selection has been shown to perform surprisingly well [38, 116], but other more principled dimensionality reductions such as latent dirichlet allocation [16] could also be used to reduce the feature set size.

Table 8.1 shows the efficiency of parallel EM for various reasonable feature set sizes. Note that as the feature set increases in size, the amount of communication increases at a faster rate than the amount of computation which must be performed for each task, which decreases parallel efficiency. In fact, if one were to do no feature engineering whatsoever and use all 80,000 words as features, the cost of writing this large number of features is so high, that when using a distributed filesystem, the serial implementation of EM runs nearly twice as fast as the parallel version. However, that is not the point here, rather we show that in this application, for any reasonable number of features, eliminating the use of redundant storage significantly improves performance. In addition, rare convergence checks in combination with our reduce-map operation brought runtime down from an average of 83.93 seconds per iteration to only 3.41 seconds, a 95.9% improvement.

8.5.2 Asynchronous MapReduce

The asynchronous programming model of Section 8.4 allows asynchronous parallel PSO [55, 109] to be expressed in MapReduce. This variant of PSO is particularly well-suited for functions whose execution time has high variance, with heterogeneous processors, and in

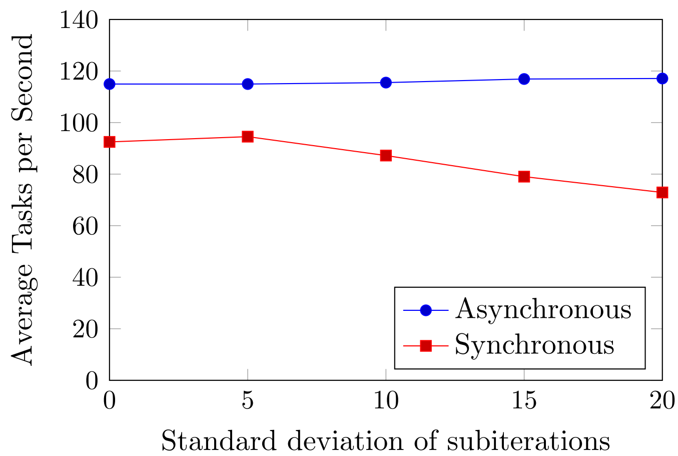


Figure 8.6: The average throughput (in tasks per second) for synchronous and asynchronous PSO. The number of subiterations per map task vary, with an average of 50 and a standard deviation ranging from 0 to 20. Throughput of the asynchronous implementation is unaffected by task variance and is better even when there is no variance.

distributed environments [100]. To evaluate the behavior of asynchronous parallel PSO in MapReduce, we vary the number of subiterations performed in each map task.

With a varying number of subiterations, asynchronous parallel PSO is distinctly faster than standard parallel PSO. We draw the number of subiterations from a normal distribution with a mean of 50 and a standard deviation ranging from 0 (no variability) to 20. Figure 8.6 shows the difference in throughput between synchronous and asynchronous PSO in MapReduce as the standard deviation varies. The throughput of asynchronous PSO is fairly constant at around 115 tasks per second. Synchronous PSO, on the other hand, slows as the standard deviation increases, with a throughput of 73 tasks per second when the standard deviation is 20.

Even with small or no standard deviation, asynchronous parallel PSO outperforms the synchronous variant. With a standard deviation of 5, synchronous PSO with combined reduce-map tasks requires an average of 0.58 seconds per iteration, while asynchronous PSO requires only 0.44 seconds. Note that reduce-map operations provide a similar benefit with variance as it does without variance: with separated reduce and map tasks, the time per iteration for synchronous PSO rises to 0.82 seconds.

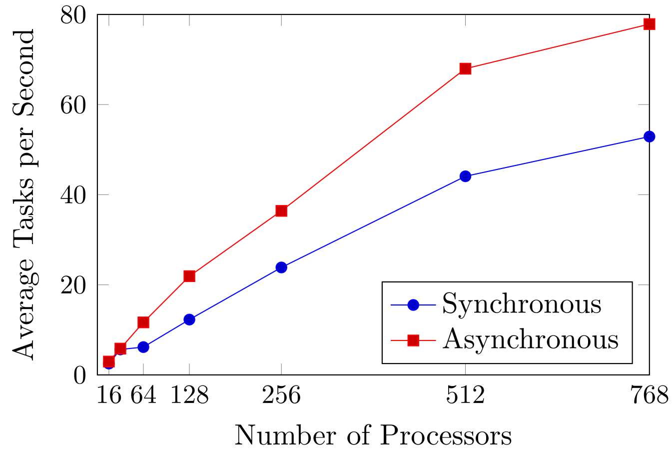


Figure 8.7: The average throughput (in tasks per second) for synchronous and asynchronous PSO with respect to the number of processors. The number of subswarms is equal to the number of processors, and the number of subiterations is 1000.

We speculate that the advantage of Asynchronous MapReduce in the case where task times are uniform is due to a more even load on the master. With synchronous MapReduce, as soon as the last task in a dataset completes, the master is suddenly able to make assignments to each of the slaves. This creates a bottleneck, not only in the master as it makes assignments, but also in the slaves as they all start communicating at the same time. In Asynchronous MapReduce, the master has no such bottleneck because it can make an assignment as soon as a single task completes, without waiting for all other tasks in the dataset to finish. Figure 8.7 explores this phenomenon and shows that the effect increases with the number of processors.

8.6 Conclusion

This paper takes the following approaches to make Mrs more appropriate for computationally intensive iterative algorithms:

- Checkpointing: we combine direct task-to-task communication with strategic use of a distributed filesystem to improve performance while preserving fault tolerance.

- The reduce-map operation: this operation is a combination of the reduce and map tasks which span successive iterations. It eliminates unnecessary communication and scheduling latency.
- Fully asynchronous operation: iterative algorithms which are naturally expressed in terms of asynchronous message passing can now be easily expressed and efficiently run.

These approaches have been previously shown to improve performance of parallelized iterative algorithms, and we have shown that they do the same in Mrs. Further, we add an additional approach novel to our implementation:

- A generator-callback model for task management: This model provides for both greater flexibility in the scheduling of tasks and better supports operations typically found in iterative programs, such as convergence checking being scheduled less frequently and outside of the regular MapReduce iterations.

These approaches improve the efficiency of Mrs MapReduce for all iterative algorithms but also makes it feasible for a wide range of applications where its overhead was previously too high to be practical.

Conclusions

We now conclude, summarizing the major contributions of this dissertation, as well as suggesting potential avenues of future work in the space of fine-grained topic modeling.

Chapter 9

Conclusions

We have set out to show that fine-grained topic modeling is not only possible with anchor-based topic models, but enables new topic-based applications which were not feasible using traditional probabilistic topic modeling. We first review our contributions towards this aim in this dissertation in Section 9.1, and then in Section 9.2 we discuss possible avenues of future research using this work.

9.1 Contributions

We view the problem of fine-grained topic modeling as two fold: improving the quality of the topics generated by anchor-based topic models, and improving the quality of the local topic assignments. In Part I, we propose two ways to ensure that the anchors for each topic better reflect the underlying data. In Part II, we determine how to evaluate local topic quality, and explore topic assignment strategies for fine-grained topic models. Using the results from the previous two parts, in Part III we apply fine-grained topic modeling to the problem of automatic cross-reference generation. Finally, in Part IV we explore the parallelization of anchor-based topic models, demonstrating significant speedup on both topic recovery and cooccurrence matrix construction.

Our main contribution towards improved anchor selection is the concept of tandem anchoring. Ordinarily, each topic is anchored by a single word which uniquely identifies the topic. However, with tandem anchors, we are able to create anchors out of multiple words which jointly express a concept. This allows us to create topics which are anchored by entire documents, which in turn allows us to create more nuanced and refined topics compared to the default single-word anchors from the Gram-Schmidt process.

In addition to tandem anchoring, we also develop the technique of labeled anchors, which allows the anchor selection process and topic recovery procedure to be influenced by document metadata values. This is useful for topic-based document classification, but also in cases where fine-grained topics can be improved by incorporating information from metadata.

We also make efforts towards improving the accuracy of token-level topic assignments. Our main contribution in this space was to develop the word topic matching task, which allows us to get user input on local topic model quality. We use this task to develop topic consistency, a metric which measures how often a model switches between topics, and we demonstrate that topic consistency correlates well with human evaluations of the quality of topic assignments. Using this result, we are able to explore topic assignment strategies. We report a mixed result on this front, with iterated conditional modes with a per-word initialization yielding the most consistent topic assignments, while mean field variational inference produces the best topic-based classification results.

Using these results, we are able to demonstrate that fine-grained topic modeling can be applied to the task of automatic cross-reference generation. This system predicts cross-references with sufficient precision to dramatically lower the cost of producing cross-reference resources for new texts. We note that this cost-savings is not possible with traditional coarse-grained topic modeling.

9.2 Future Work

Having validated our thesis statement that fine-grained topic modeling is not only possible using anchor-based methods, but demonstrated that it can be applied to the problem of cross-reference generation, we suggest the future work in the space of fine-grained topic modeling could focus on exploring other topic-based applications for which coarse-grained topic modeling is inadequate.

One such application is the problem of automatic redaction, in which sensitive phrases or sentences are marked as confidential. Topic-based classification can classify entire documents [96], but with coarse-grained topics, it can be difficult to have a small number of topics cover every aspect of sensitive subjects which should be redacted. We submit that with fine-grained topic modeling, a topic-based approach might be suitable for this problem.

A related problem which has received a lot of attention in recent literature is aspect level sentiment classification [98]. Rather than trying to identify overall sentiment polarity, aspect level sentiment classification attempts to identify sentiment with respect to a specific context. For example, in the sentence "we loved the atmosphere, the food, and the service, but it was bit pricey", the overall sentiment might be positive, even though for the aspect of price the sentiment is more negative. We believe that a topic-based solution using labeled anchors and fine-grained topics is a promising approach to solving this problem.

These are just two potential applications for fine-grained topic modeling, but undoubtedly there are many other areas that this work could improve. Furthermore, future work into additional applications can serve as a springboard spurring further development of topic modeling research.

Appendix A

Relationship Between Topic Models and Word Embedding Models

On a surface level, topic modeling and word embedding models are very closely related. Both types of algorithm seek to explain the cooccurrences between words in a low-dimension space. There are however, differences in how the algorithms are trained. At the core, topic modeling is about documents, and word embeddings are about words.

More specifically, word embedding models explain words by their contexts. For example, using the popular Word2Vec [75] family of word embeddings with a continuous bag of words representation of the data, each word w is modeled as

$$p(w|w_1, w_2, ..., w_n) \propto exp(v_w \cdot (\frac{1}{n}\sum_{i}^{n} v_{w_i})) \tag{A.1}$$

where w_i is the ith word of the context of w, and v_w gives the vector representation of word w. Assuming that we have sufficient data to estimate the left-hand side of Equation A.1, the problem of finding the vectors v on the right-hand side is essentially a non-convex optimization problem which can be solved with the aid of the negative sampling trick introduced by Mikolov et al. [75].

There are many variants of word embedding models, including the count-based GloVe model [90], but the unifying characteristic is that they all seek to represent words in vector-space based on the contexts they appear in. A key point to notice is that these algorithms ignore documents and document boundaries.

In contrast, topic modeling is based on word cooccurrences within documents. Topic models typically assume that each document has a distribution of topics, and that the words of the document come from those topics. In other words, each word w is modeled as

$$p(w|d) = \sum_{z}^{K} p(w|z)p(z|d) \tag{A.2}$$

where z is one of K topics, and d is the document containing w. While the specific distributions differ depending on the model, the important thing is that the inference of the topics, or the topic-word distributions $p(w|z)$ depends on cooccurrences between words within documents.

If the V word types of a corpus are represented in a V-dimensional cooccurrence space, both word embeddings and topic models seek to compress these representations into a K-dimensional space which preserves pairwise distances and for which $K \ll V$. In fact, the anchor word algorithm [6] does this explicitly, using non-negative matrix factorization to compute a $V \times K$ word-topic matrix which represents this projection. However, the results of the two classes of algorithm are quite different.

With word embeddings, the dimensions of the compressed K-dimensional space typically represent the meanings of words. It is for this reason that we can use word embeddings to solve analogies using simple linear relationships between vector representations of words. For example, if find the word w which minimizes

$$\|v_w - v_{king} + v_{man} - v_{woman}\|^2 \tag{A.3}$$

many word embeddings would result in the word "queen", which is an appropriate answer giving us the analogy man:woman::king:queen.

Recent work in word embeddings has used this linear algebraic structure of meaning captured by word embeddings for word sense induction [7]. This work sets up an optimization problem in which words are decomposed into 'discourse atoms', which are vectors in the word embedding space. In many respects, these discourse atoms are very similar to topics in that cosine distance between word vectors and the atoms produces meaningful distributions over words.

Atom 1978	825	231	616	1638	149	330
drowning	instagram	stakes	membrane	slapping	orchestra	conferences
suicides	twitter	thoroughbred	mitochondria	pulling	philharmonic	meetings
overdose	facebook	guineas	cytosol	plucking	philharmonia	seminars
murder	tumblr	preakness	cytoplasm	squeezing	conductor	workshops
poisoning	vimeo	filly	membranes	twisting	symphony	exhibitions
commits	linkedin	fillies	organelles	bowing	orchestras	organizes
stabbing	reddit	epsom	endoplasmic	slamming	toscanini	concerts
strangulation	myspace	racecourse	proteins	tossing	concertgebouw	lectures
gunshot	tweets	sired	vesicles	grabbing	solti	presentations

Figure A.1: Some discourse atoms and their nearest 9 words. Each atom is labeled with an arbitrary id number, and the word embeddings are trained on 3 billion tokens of Wikipedia data. Figure taken from Arora et al. [7].

Figure A.1, taken from Arora et al. [7], shows the result of training a word embedding on 3 billion tokens of Wikipedia data and using this method of computing discourse atoms. The closest 9 words from each atom are shown. Note that the words associated with each atom are based on meaning. Because there are 2000 such atoms, the meanings associated with each atom can be fairly nuanced and subtle.

For example, atom 825 includes words such as 'instagram', 'twitter' and 'facebook'. On the surface, this may seem like a topic from a typical topic model dealing with social media. Indeed, if a corpus had multiple documents discussing various social media platforms in such a way that these word frequently cooccurred within the same documents, such a topic would be possible. However, this is not how this particular discourse atom came to be.

Instead, this discourse atom was recovered due to the fact that these words often appear in the similar contexts. For example, for the context 'I posted the photo on _____', the terms for these social media platforms are largely interchangeable, and so word embeddings minimize the distance between the vectors for these words.

In contrast, consider the topics in Figure A.2 as represented by their top n most probable words. These topics are produced using the anchor-words algorithm [6] on the Twenty Newsgroups dataset[1]. Note that unlike the atoms in Figure A.1, the topics of

[1]These topics are reused from Chapter 2, which deals with tandem anchoring. However, these are topics produced using single word anchors.

alt.atheism	talk.religion.misc	rec.baseball	rec.hockey
evolution	religion	baseball	hockey
theory	god	games	team
science	government	players	play
faith	state	word	games
quote	jesus	teams	season
facts	israel	car	players

Figure A.2: Some topics represented by the 6 most probable words in a topic. We label each topic using the newsgroup the topic is most heavily associated with. The topics are learned using the anchor-word algorithm on the Twenty Newsgroups dataset.

Figure A.2 do not focus on the meaning of words, but instead capture themes or concepts that appear in the documents of the Twenty Newsgroups dataset.

For example, consider topic associated with the baseball newsgroup. The words of this topic, such as 'teams' and 'games' and 'baseball' are all clearly related and used together frequently as part of the discussion about the concept of baseball. However, these words do not necessarily have the same meaning, nor are the necessarily interchangeable within the same contexts. This topic is recovered not because of similar meaning, but because of frequent cooccurence within documents discussing baseball.

Besides the difference in the interpretation of the compressed representation of words in word embeddings and topic modeling, the document-based view of words taken by topic modeling allows us to produce topic assignments for individual documents and word, while word embeddings only provide the mapping of words to meaning. These topic assignments allow us to summarize the topical content of documents, without paying attention to the meaning of the individual words. Word embeddings do essentially the opposite, focusing on the meaning of each individual word in isolation.

Because of these differences, topic models and word embeddings are used for different tasks. Essentially, topic models are about documents. Topic modeling seeks to remove individual words from the view of documents, and instead lets us summarize entire documents by their topical content. Word embeddings on the other hand are about words. Rather than replace words with a high-level topic, we replace each word with a vector which indicates a specific meaning for each individual word.

The applications of these two types of models are therefore very different. For example, in natural language processing word embeddings are commonly used as the first layer in neural models designed for tasks like machine translation. Because topic model output actually conflates words with different meanings based on document cooccurrences, topic models would likely be a poor word representation for these tasks. On the other hand, for tasks such as topic-based cross-referencing (see Chapter 6), we need to know what documents are about regardless of the meaning of the individual words used, so word embeddings would likely not be as useful as topic models in this case.

With these differences in mind, we note that this dissertation is concerned with topic modeling and applications of fine-grained topic models. Consequently, while the literature regarding word embeddings is rich, this dissertation avoids the discussion of word embeddings, as it is not strictly comparable or applicable line of related work.

Appendix B

Examples of Biblical Cross-References

In this appendix we give examples of cross-references for the English Standard Version of the Bible discovered by the system described in Chapter 6. The rank of each reference is determined using a model with 3,000 tandem anchors and using cosine distance to compare document-topic vectors. We show verse pairs from various rankings to give the reader a sense of the types of cross-references found by our system.

The first column indicates how the model ranked a particular verse pair. The second column indicates whether the predicted cross-reference was valid according to the Treasury of Scripture Knowledge, Enhanced. The third and fourth column indicates whether the predicted cross-reference was valid according to OpenBible+0 and OpenBible+5 respectively. For these columns, an 'X' indicates that the verse pair was valid under the cross-reference dataset, while a '-' indicates that the verse pair was not found in the given cross-reference dataset. We then give verse and verse text of the source verse, and finally the verse and verse text of the target verse.

Rank	TSKE	OB+0	OB+5	Source	Target
4	-	-	-	1Sam.30.30 in Hormah, in Borashan, in Athach,	Josh.15.30 Eltolad, Chesil, Hormah,
21	X	-	-	Gen.11.12 When Arpachshad had lived 35 years, he fathered Shelah.	Gen.10.24 Arpachshad fathered Shelah; and Shelah fathered Eber.
23	X	X	-	Gen.10.24 Arpachshad fathered Shelah; and Shelah fathered Eber.	Gen.11.12 When Arpachshad had lived 35 years, he fathered Shelah.
676	X	X	X	Prov.16.25 There is a way that seems right to a man, but its end is the way to death.	Prov.14.12 There is a way that seems right to a man, but its end is the way to death.
1000	-	-	-	Gen.36.37 Samlah died, and Shaul of Rehoboth on the Euphrates reigned in his place.	Gen.36.36 Hadad died, and Samlah of Masrekah reigned in his place.
1007	X	-	-	Job.25.1 Then Bildad the Shuhite answered and said:	Job.18.1 Then Bildad the Shuhite answered and said:
1011	X	X	-	2Kgs.25.20 And Nebuzaradan the captain of the guard took them and brought them to the king of Babylon at Riblah.	Jer.52.26 And Nebuzaradan the captain of the guard took them and brought them to the king of Babylon at Riblah.
1128	X	X	X	Matt.6.27 And which of you by being anxious can add a single hour to his span of life?	Luke.12.25 And which of you by being anxious can add a single hour to his span of life?
2000	-	-	-	1Chr.8.5 Gera, Shephuphan, and Huram.	1Chr.8.7 Naaman, Ahijah, and Gera, that is, Heglam, who fathered Uzza and Ahihud.

Rank	TSKE	OB+0	OB+5	Source	Target
2005	X	-	-	Rom.12.4 For as in one body we have many members, and the members do not all have the same function,	Rom.12.5 so we, though many, are one body in Christ, and individually members one of another.
2006	X	X	-	Eph.5.30 because we are members of his body.	Rom.12.5 so we, though many, are one body in Christ, and individually members one of another.
2021	X	X	X	Ps.57.5 Be exalted, O God, above the heavens! Let your glory be over all the earth!	Ps.57.11 Be exalted, O God, above the heavens! Let your glory be over all the earth!
5000	-	-	-	Exod.38.15 And so for the other side. On both sides of the gate of the court were hangings of fifteen cubits, with their three pillars and their three bases.	Exod.27.15 On the other side the hangings shall be fifteen cubits, with their three pillars and three bases.
5007	X	-	-	Exod.39.2 He made the ephod of gold, blue and purple and scarlet yarns, and fine twined linen.	Exod.28.5 They shall receive gold, blue and purple and scarlet yarns, and fine twined linen.
5011	X	X	-	Neh.7.44 The singers: the sons of Asaph, 148.	1Chr.25.2 Of the sons of Asaph: Zaccur, Joseph, Nethaniah, and Asharelah, sons of Asaph, under the direction of Asaph, who prophesied under the direction of the king.

Rank	TSKE	OB+0	OB+5	Source	Target
5242	X	X	X	Matt.12.30 Whoever is not with me is against me, and whoever does not gather with me scatters.	Luke.11.23 Whoever is not with me is against me, and whoever does not gather with me scatters.
10000	-	-	-	Ezek.40.8 Then he measured the vestibule of the gateway, on the inside, one reed.	Ezek.42.17 He measured the north side, 500 cubits by the measuring reed all around.
10016	X	-	-	Matt.13.7 Other seeds fell among thorns, and the thorns grew up and choked them.	Mark.4.7 Other seed fell among thorns, and the thorns grew up and choked it, and it yielded no grain.
10017	X	X	-	Mark.4.7 Other seed fell among thorns, and the thorns grew up and choked it, and it yielded no grain.	Matt.13.7 Other seeds fell among thorns, and the thorns grew up and choked them.
10130	X	X	X	Matt.25.23 His master said to him, 'Well done, good and faithful servant. You have been faithful over a little; I will set you over much. Enter into the joy of your master.'	Matt.25.21 His master said to him, 'Well done, good and faithful servant.You have been faithful over a little; I will set you over much. Enter into the joy of your master.'
20000	-	-	-	Num.33.37 And they set out from Kadesh and camped at Mount Hor, on the edge of the land of Edom.	Num.33.41 And they set out from Mount Hor and camped at Zalmonah.

Rank	TSKE	OB+0	OB+5	Source	Target
20016	X	-	-	2Chr.26.4 And he did what was right in the eyes of the Lord, according to all that his father Amaziah had done.	2Kgs.15.34 And he did what was right in the eyes of the Lord, according to all that his father Uzziah had done.
20017	X	X	-	2Kgs.15.34 And he did what was right in the eyes of the Lord, according to all that his father Uzziah had done.	2Chr.26.4 And he did what was right in the eyes of the Lord, according to all that his father Amaziah had done.
20578	X	X	X	Eph.5.19 addressing one another in psalms and hymns and spiritual songs, singing and making melody to the Lord with your heart,	Col.3.16 Let the word of Christ dwell in you richly, teaching and admonishing one another in all wisdom, singing psalms and hymns and spiritual songs, with thankfulness in your hearts to God.
110210	-	-	-	Prov.21.21 Whoever pursues righteousness and kindness will find life, righteousness, and honor.	John.3.16 "For God so loved the world,that he gave his only Son, that whoever believes in him should not perish but have eternal life.

rank	tske	ob+0	ob+5	source	target
110230	X	-	-	Luke.8.46 But Jesus said, "Someone touched me, for I perceive that power has gone out from me."	Mark.5.30 And Jesus, perceiving in himself that power had gone out from him, immediately turned about in the crowd and said, "Who touched my garments?"
110231	X	X	-	Mark.5.30 And Jesus, perceiving in himself that power had gone out from him, immediately turned about in the crowd and said, "Who touched my garments?"	Luke.8.46 But Jesus said, "Someone touched me, for I perceive that power has gone out from me."
111402	X	X	X	Luke.3.16 John answered them all, saying, "I baptize you with water, but he who is mightier than I is coming, the strap of whose sandals I am not worthy to untie. He will baptize you with the Holy Spirit and fire.	John.1.33 I myself did not know him, but he who sent me to baptize with water said to me, 'He on whom you see the Spirit descend and remain, this is he who baptizes with the Holy Spirit.'

Rank	TSKE	OB+0	OB+5	Source	Target
221230	-	-	-	Acts.13.31 and for many days he appeared to those who had come up with him from Galilee to Jerusalem, who are now his witnesses to the people.	Prov.25.18 A man who bears false witness against his neighbor is like a war club, or a sword, or a sharp arrow.
221331	X	-	-	2Chr.22.8 And when Jehu was executing judgment on the house of Ahab, he met the princes of Judah and the sons of Ahaziah's brothers, who attended Ahaziah, and he killed them.	2Chr.22.9 He searched for Ahaziah, and he was captured while hiding in Samaria, and he was brought to Jehu and put to death. They buried him, for they said, "He is the grandson of Jehoshaphat, who sought the Lord with all his heart." And the house of Ahaziah had no one able to rule the kingdom.
221364	X	X	-	Hag.1.15 on the twenty-fourth day of the month, in the sixth month, in the second year of Darius the king.	Hag.2.20 The word of the Lord came a second time to Haggai on the twenty-fourth day of the month,

Rank	TSKE	OB+0	OB+5	Source	Target
221548	X	X	X	Jonah.1.17 And the Lord appointed a great fish to swallow up Jonah. And Jonah was in the belly of the fish three days and three nights.	Matt.12.40 For just as Jonah was three days and three nights in the belly of the great fish, so will the Son of Man be three days and three nights in the heart of the earth.
316739	-	-	-	Job.29.24 I smiled on them when they had no confidence, and the light of my face they did not cast down.	Joel.2.6 Before them peoples are in anguish; all faces grow pale.
316772	X	-	-	Amos.4.5 offer a sacrifice of thanksgiving of that which is leavened, and proclaim freewill offerings, publish them; for so you love to do, O people of Israel!" declares the Lord God.	Lev.7.13 With the sacrifice of his peace offerings for thanksgiving he shall bring his offering with loaves of leavened bread.
316773	X	X	-	Lev.7.13 With the sacrifice of his peace offerings for thanksgiving he shall bring his offering with loaves of leavened bread.	Amos.4.5 offer a sacrifice of thanksgiving of that which is leavened, and proclaim freewill offerings, publish them; for so you love to do, O people of Israel!" declares the Lord God.

Rank	TSKE	OB+0	OB+5	Source	Target
317001	X	X	X	John.12.49 For I have not spoken on my own authority, but the Father who sent me has himself given me a commandment—what to say and what to speak.	John.14.10 Do you not believe that I am in the Father and the Father is in me? The words that I say to you I do not speak on my own authority, but the Father who dwells in me does his works.
536739	-	-	-	Num.9.19 Even when the cloud continued over the tabernacle many days, the people of Israel kept the charge of the Lord and did not set out.	Ezek.44.17 When they enter the gates of the inner court, they shall wear linen garments. They shall have nothing of wool on them, while they minister at the gates of the inner court, and within.
536757	X	-	-	Mark.12.26 And as for the dead being raised, have you not read in the book of Moses, in the passage about the bush, how God spoke to him, saying, 'I am the God of Abraham, and the God of Isaac, and the God of Jacob'?	Exod.3.6 And he said, "I am the God of your father, the God of Abraham, the God of Isaac, and the God of Jacob." And Moses hid his face, for he was afraid to look at God.

Rank	TSKE	OB+0	OB+5	Source	Target
536774	X	X	-	2Sam.5.12 And David knew that the Lord had established him king over Israel, and that he had exalted his kingdom for the sake of his people Israel.	1Kgs.10.9 Blessed be the Lord your God, who has delighted in you and set you on the throne of Israel! Because the Lord loved Israel forever, he has made you king, that you may execute justice and righteousness."
537418	X	X	X	Mark.8.18 Having eyes do you not see, and having ears do you not hear? And do you not remember?	Ezek.12.2 "Son of man, you dwell in the midst of a rebellious house, who have eyes to see, but see not, who have ears to hear, but hear not, for they are a rebellious house.
1536739	-	-	-	2Chr.5.10 There was nothing in the ark except the two tablets that Moses put there at Horeb, where the Lord made a covenant with the people of Israel, when they came out of Egypt.	Josh.9.11 So our elders and all the inhabitants of our country said to us, 'Take provisions in your hand for the journey and go to meet them and say to them, "We are your servants. Come now, make a covenant with us."'

Rank	TSKE	OB+0	OB+5	Source	Target
1536828	X	-	-	Judg.1.30 Zebulun did not drive out the inhabitants of Kitron, or the inhabitants of Nahalol, so the Canaanites lived among them, but became subject to forced labor.	Deut.20.11 And if it responds to you peaceably and it opens to you, then all the people who are found in it shall do forced labor for you and shall serve you.
1536829	X	X	-	Deut.20.11 And if it responds to you peaceably and it opens to you, then all the people who are found in it shall do forced labor for you and shall serve you.	Judg.1.30 Zebulun did not drive out the inhabitants of Kitron, or the inhabitants of Nahalol, so the Canaanites lived among them, but became subject to forced labor.
1537240	X	X	X	Prov.21.19 It is better to live in a desert land than with a quarrelsome and fretful woman.	Prov.21.9 It is better to live in a corner of the housetop than in a house shared with a quarrelsome wife.
10234713	-	-	-	Num.2.33 But the Levites were not listed among the people of Israel, as the Lord commanded Moses.	2Chr.19.7 Now then, let the fear of the Lord be upon you. Be careful what you do, for there is no injustice with the Lord our God, or partiality or taking bribes."

Rank	TSKE	OB+0	OB+5	Source	Target
10234763	X	-	-	Jer.50.2 "Declare among the nations and proclaim, set up a banner and proclaim, conceal it not, and say: 'Babylon is taken, Bel is put to shame, Merodach is dismayed. Her images are put to shame, her idols are dismayed.'	Jer.50.38 A drought against her waters, that they may be dried up! For it is a land of images, and they are mad over idols.
10234824	X	X	-	Prov.19.11 Good sense makes one slow to anger, and it is his glory to overlook an offense.	Matt.5.44 But I say to you, Love your enemies and pray for those who persecute you,
10243421	X	X	X	Deut.12.20 "When the Lord your God enlarges your territory, as he has promised you, and you say, 'I will eat meat,' because you crave meat, you may eat meat whenever you desire.	Deut.19.8 And if the Lord your God enlarges your territory, as he has sworn to your fathers, and gives you all the land that he promised to give to your fathers—

References

[1] Loulwah AlSumait, Daniel Barbará, James Gentle, and Carlotta Domeniconi. Topic significance ranking of LDA generative models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 67–82, 2009.

[2] Lavina Fielding Anderson. Church publishes first LDS edition of the Bible. *Ensign*, page 9, October 1979.

[3] David Andrzejewski, Xiaojin Zhu, and Mark Craven. Incorporating domain knowledge into topic modeling via Dirichlet forest priors. In *Proceedings of the International Conference of Machine Learning*, pages 25–32, 2009.

[4] Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. Computing a nonnegative matrix factorization–provably. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 145–162, 2012.

[5] Sanjeev Arora, Rong Ge, and Ankur Moitra. Learning topic models–going beyond SVD. In *Fifty-Third IEEE Annual Symposium on Foundations of Computer Science*, pages 1–10, 2012.

[6] Sanjeev Arora, Rong Ge, Yonatan Halpern, David Mimno, Ankur Moitra, David Sontag, Yichen Wu, and Michael Zhu. A practical algorithm for topic modeling with provable guarantees. In *Proceedings of the International Conference of Machine Learning*, pages 280–288, 2013.

[7] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association of Computational Linguistics*, 6:483–495, 2018.

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

[9] Georgios Balikas, Massih-Reza Amini, and Marianne Clausel. On a topic model for sentences. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 921–924, 2016.

[10] Georgios Balikas, Hesam Amoualian, Marianne Clausel, Eric Gaussier, and Massih-Reza Amini. Modeling topic dependencies in semantically coherent text spans with copulas. In *Proceedings of International Conference on Computational Linguistics*, 2016.

[11] Julian Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 259–302, 1986.

[12] Crossway bibles. *The Holy Bible: English Standard Version*. Crossway Bibles, 2001.

[13] David M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, April 2012.

[14] David M Blei and John D Lafferty. A correlated topic model of science. *The Annals of Applied Statistics*, pages 17–35, 2007.

[15] David M Blei and Jon D McAuliffe. Supervised topic models. *arXiv preprint arXiv:1003.0783*, 2010.

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

[17] Jordan L Boyd-Graber and David M Blei. Syntactic topic models. In *Proceedings of Advances in Neural Information Processing Systems*, pages 185–192, 2009.

[18] Jordan L Boyd-Graber, David M Blei, and Xiaojin Zhu. A topic model for word sense disambiguation. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1138–1147, 2007.

[19] D. Bratton and J. Kennedy. Defining a standard for particle swarm optimization. In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 120–127, 2007.

[20] Y. Bu, B. Howe, M. Balazinska, and M. Ernst. HaLoop: Efficient iterative data processing on large clusters. *Proceedings of the VLDB Endowment*, 3(1-2):285–296, 2010.

[21] C. Chambers, A. Raniwala, F. Perry, S. Adams, R.R. Henry, R. Bradshaw, and N. Weizenbaum. FlumeJava: Easy, efficient data-parallel pipelines. In *Proceedings of the ACM SIGPLAN conference of programming language design and implementation*, pages 363–375, 2010.

[22] Allison June-Barlow Chaney and David M Blei. Visualizing topic models. In *ICWSM*, 2012.

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

[24] Jaegul Choo, Changhyun Lee, Chandan K Reddy, and Heejung Park. Utopian: User-driven topic modeling based on interactive nonnegative matrix factorization. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):1992–2001, 2013.

[25] C.T. Chu, S.K. Kim, Y.A. Lin, Y.Y. Yu, G. Bradski, A.Y. Ng, and K. Olukotun. Map-Reduce for machine learning on multicore. In *Advances in Neural Information Processing Systems*, pages 281–288, 2007.

[26] Jason Chuang, Christopher D Manning, and Jeffrey Heer. Termite: Visualization techniques for assessing textual topic models. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 74–77, 2012.

[27] Jason Chuang, Daniel Ramage, Christopher Manning, and Jeffrey Heer. Interpretation and trust: Designing model-driven visualizations for text analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 443–452. ACM, 2012.

[28] Kristin A. Cook and James J. Thomas. Illuminating the path: The research and development agenda for visual analytics. Technical report, Pacific Northwest National Laboratory (PNNL), Richland, WA (US), 2005.

[29] Ralph D'Agostino and Egon S Pearson. Tests for departure from normality. empirical results for the distributions of b2 and $\sqrt{b1}$. *Biometrika*, 60(3):613–622, 1973.

[30] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[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, 1990.

[32] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1), 1977.

[33] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[34] David Donoho and Victoria Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *Proceedings of Advances in Neural Information Processing Systems*, pages 1141–1148, 2003.

[35] Jacob Eisenstein, Amr Ahmed, and Eric P Xing. Sparse additive generative models of text. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1041–1048, 2011.

[36] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.H. Bae, J. Qiu, and G. Fox. Twister: a runtime for iterative MapReduce. In *High Performance Distributed Computing*, pages 810–818, 2010.

[37] E. Elnikety, T. Elsayed, and H.E. Ramadan. iHadoop: Asynchronous iterations for mapreduce. In *Proceedings of the IEEE Cloud Computing Technology and Science*, pages 81–90, 2011.

[38] Kuzman Ganchev and Mark Dredze. Small statistical models by random feature mixing. In *Workshop on Mobile NLP*, pages 19–20, 1998.

[39] Matthew J Gardner, Joshua Lutes, Jeff Lund, Josh Hansen, Dan Walker, Eric Ringger, and Kevin Seppi. The topic browser: An interactive tool for browsing topic models. In *NIPS Workshop on Challenges of Data Visualization*, 2010.

[40] Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar. *Introduction to Parallel Computing*. Addison-Wesley, Harlow, England, second edition, 2003.

[41] DMBTL Griffiths and MIJJB Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. *Advances in neural information processing systems*, 16: 17–24, 2004.

[42] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, 2004.

[43] Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. Integrating topics and syntax. In *Advances in neural information processing systems*, pages 537–544, 2004.

[44] Guiness. Best selling book of non-fiction, October 2018. http://www.guinnessworldrecords.com/world-records/best-selling-book-of-non-fiction.

[45] Joshua A Hansen, Eric K Ringger, and Kevin D Seppi. Probabilistic explicit topic modeling using wikipedia. In *Language Processing and Knowledge in the Web*, pages 69–82. Springer, 2013.

[46] Harold Stanley Heaps. *Information retrieval: Computational and theoretical aspects*, pages 206–208. Academic Press, Inc., 1978.

[47] Matthew Hoffman, Francis R Bach, and David M Blei. Online learning for latent Dirichlet allocation. In *advances in neural information processing systems*, pages 856–864, 2010.

[48] Matthew D Hoffman, David M Blei, Chong Wang, and John William Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[49] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

[50] Yuening Hu and Jordan Boyd-Graber. Efficient tree-based topic modeling. In *Proceedings of the Association for Computational Linguistics*, pages 275–279, 2012.

[51] Yuening Hu, Jordan L Boyd-Graber, and Brianna Satinoff. Interactive topic modeling. In *Proceedings of the Association for Computational Linguistics*, pages 248–257, 2011.

[52] Yuening Hu, Jordan Boyd-Graber, Brianna Satinoff, and Alison Smith. Interactive topic modeling. *Machine Learning*, 95(3):423–469, June 2014.

[53] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: Distributed data-parallel programs from sequential building blocks. *ACM SIGOPS Operating Systems Review*, 41(3), 2007.

[54] James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *International Conference on Neural Networks*, pages 1942–1948, 1995.

[55] Byung-Il Koh, Alan George, Raphael Haftka, and Benjamin Fregly. Parallel asynchronous particle swarm optimization. *International Journal of Numerical Methods in Engineering*, 67:578–595, 2006.

[56] Klaus Krippendorff. *Content Analysis: An Introduction to Its Methodology*, pages 221–250. Thousand Oaks, 3rd edition, 2013.

[57] N. Kumar, Sanjiv Satoor, and Ian Buck. Fast parallel expectation maximization for gaussian mixture models on GPUs using CUDA. In *Proc. High Performance Computing and Communications*, pages 103–109, 2009.

[58] Ken Lang. Newsweeder: Learning to filter netnews. In *International Conference on Machine Learning*, 1995.

[59] Jey Han Lau and Timothy Baldwin. The sensitivity of topic coherence evaluation to topic cardinality. In *Proceedings of the NAACL HLT 2010 Sixth Web as Corpus Workshop*, pages 483–487. Association for Computational Linguistics, 2016.

[60] Moontae Lee and David Mimno. Low-dimensional embeddings for interpretable anchor-based topic inference. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1319–1328, 2014.

[61] Wei Li and Andrew McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *Proceedings of the 23rd international conference on Machine learning*, pages 577–584. ACM, 2006.

[62] Yi-Kai Liu, Animashree Anandkumar, Dean P Foster, Daniel Hsu, and Sham M Kakade. Two svds suffice: Spectral decompositions for probabilistic topic modeling and latent Dirichlet allocation. In *Neural Information Processing Systems*, 2012.

[63] Yucheng Low, Joseph E Gonzalez, Aapo Kyrola, Danny Bickson, Carlos E Guestrin, and Joseph Hellerstein. Graphlab: A new framework for parallel machine learning. *arXiv:1408.2041*, 2014.

[64] Jeffrey Lund, Chace Ashcraft, Andrew McNabb, and Kevin Seppi. Mrs: high performance mapreduce for iterative and asynchronous algorithms in python. In *Workshop on Python for High-Performance and Scientific Computing*, pages 76–85. IEEE, 2016.

[65] Jeffrey Lund, Paul Felt, Kevin Seppi, and Eric K. Ringger. Fast inference for interactive models of text. In *Proceedings of the 26th International Conference on Computational Linguistics*, pages 2997–3006. Association for Computational Linguistics, 2016.

[66] Jeffrey Lund, Connor Cook, Kevin Seppi, and Jordan Boyd-Graber. Tandem anchoring: A multiword anchor approach for interactive topic modeling. In *Proceedings of the Association for Computational Linguistics*, pages 896–905, 2017.

[67] Jeffrey Lund, Stephen Cowley, Wilson Fearn, Emily Hales, and Kevin Seppi. Labeled anchors and a scalable, transparent, and interactive classifier. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 824–829, 2018.

[68] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[69] G. Mann, R. McDonald, M. Mohri, N. Silberman, and D. Walker. Efficient large-scale distributed training of conditional maximum entropy models. *Advances in Neural Information Processing Systems*, pages 1231–1239, 2009.

[70] A. McNabb, J. Lund, and K. Seppi. Mrs: MapReduce for scientific computing in Python. In *Python for High Performance and Scientific Computing*, pages 600–608, 2012.

[71] Andrew McNabb, Christopher Monson, and Kevin Seppi. MRPSO: MapReduce particle swarm optimization. In *Conference on Genetic and Evolutionary Computation*, 2007.

[72] Andrew McNabb, Jeffrey Lund, and Kevin Seppi. Mrs: Mapreduce for scientific computing in python. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, pages 600–608. IEEE, 2012.

[73] Marina Meilă. Comparing clusterings by the variation of information. In *Learning theory and kernel machines*, pages 173–187. Springer, 2003.

[74] Marina Meila and David Heckerman. An experimental comparison of model-based clustering methods. *Machine Learning*, pages 1–2, 2001.

[75] 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*, pages 3111–3119, 2013.

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

[77] Thomas Minka and John Lafferty. Expectation-propagation for the generative aspect model. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 352–359. Morgan Kaufmann Publishers Inc., 2002.

[78] Thomas P Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc., 2001.

[79] Girjesh Mishra, Shraddha Masih, Sanjay Tanwani, and Mohan Bansal. Glister: A framework for iterative mapreduce. In *International Conference on Computer, Communication and Control*, pages 1–6. IEEE, 2015.

[80] Timothy Morton. *Treasury of Scripture Knowledge, Enhanced.* BibleAnalyzer.com, 2010.

[81] Kevin P Murphy. *Machine learning: a probabilistic perspective*, pages 928–929. MIT Press, 2012.

[82] D.G. Murray, M. Schwarzkopf, C. Smowton, S. Smith, A. Madhavapeddy, and S. Hand. Ciel: a universal execution engine for distributed data-flow computing. In *Proceedings of Network Systems Design and Implementation*, pages 113–126, 2011.

[83] David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. Automatic evaluation of topic coherence. In *Proceedings of the Association for Computational Linguistics*, pages 100–108, 2010.

[84] David Newman, Edwin V Bonilla, and Wray Buntine. Improving topic coherence with regularized topic models. In *Proceedings of Advances in Neural Information Processing Systems*, pages 496–504, 2011.

[85] Thang Nguyen, Yuening Hu, and Jordan L Boyd-Graber. Anchors regularized: Adding robustness and extensibility to scalable topic-modeling algorithms. In *Proceedings of the Association for Computational Linguistics*, pages 359–369, 2014.

[86] Thang Nguyen, Jordan Boyd-Graber, Jeffrey Lund, Kevin Seppi, and Eric Ringger. Is your anchor going up or down? Fast and accurate supervised topic models. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 746–755, 2015.

[87] Stefanie Nowak and Stefan Rüger. How reliable are annotations via crowdsourcing: a study about inter-annotator agreement for multi-label image annotation. In *Proceedings of the international conference on Multimedia information retrieval*, pages 557–566. ACM, 2010.

[88] Christos H Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 159–168. ACM, 1998.

[89] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[90] 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. URL `http://www.aclweb.org/anthology/D14-1162`.

[91] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D Manning. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 248–256. Association for Computational Linguistics, 2009.

[92] Joseph Reisinger, Austin Waters, Bryan Silverthorn, and Raymond J Mooney. Spherical topic models. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 903–910, 2010.

[93] Juan Romero and Carlos Cotta. Optimization by island-structured decentralized particle swarms. In *Proceedings of Fuzzy Days: Computational Intelligence, Theory and Applications*, pages 25–33, 2005.

[94] Joshua Rosen, Neoklis Polyzotis, Vinayak Borkar, Yingyi Bu, Michael J Carey, Markus Weimer, Tyson Condie, and Raghu Ramakrishnan. Iterative mapreduce for large scale machine learning. *arXiv:1303.3517*, 2013.

[95] Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. The author-topic model for authors and documents. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 487–494, 2004.

[96] Timothy Rubin, America Chambers, Padhraic Smyth, and Mark Steyvers. Statistical topic models for multi-label document classification. *Machine Learning*, 1(88):157–208, 2012.

[97] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3): e0118432, 2015.

[98] Kim Schouten and Flavius Frasincar. Survey on aspect-level sentiment analysis. *IEEE Transactions on Knowledge & Data Engineering*, (1):1–1, 2016.

[99] J. Schutte, J. Reinbolt, B. Fregly, R. Haftka, and A. George. Parallel global optimization with the particle swarm algorithm. *International Journal for Numerical Methods in Engineering*, 61(13):2296–2315, 2004.

[100] Ian Scriven, David Ireland, Andrew Lewis, Sanaz Mostaghim, and Jürgen Branke. Asynchronous multiple objective particle swarm optimisation in unreliable distributed environments. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2481–2486, 2008.

[101] Hitesh Singhal and Ram Mohana Reddy Guddeti. Modified mapreduce framework for enhancing performance of graph based algorithms by fast convergence in distributed environment. In *International Conference on Advances in Computing, Communications and Informatics*, pages 1240–1245. IEEE, 2014.

[102] Alexander Smola and Shravan Narayanamurthy. An architecture for parallel topic models. *Proceedings of the VLDB Endowment*, 3(1-2):703–710, 2010.

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

[104] James Strong. *The Exhaustive Concordance of the Bible*. Jennings & Graham, 1890.

[105] K. Tang, X. Li, P.N. Suganthan, Z. Yang, and T. Weise. Benchmark functions for the CEC'2010 special session and competition on large scale global optimization. Technical report, IEEE Congress on Evolutionary Computation, November, 2009.

[106] Ivan Titov and Ryan T McDonald. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of the Association for Computational Linguistics*, pages 308–316, 2008.

[107] R.A. Torry. *Treasury of Scripture Knowledge*. Hendrickson Publishers, 2002.

[108] W Ben Towne, Carolyn P Rosé, and James D Herbsleb. Measuring similarity similarly: Lda and human perception. *ACM Transactions on Intelligent Systems and Technology*, 8(1):7:1–7:28, 2016.

[109] Gerhard Venter and Jaroslaw Sobieszczanski-Sobieski. A parallel particle swarm optimization algorithm accelerated by asynchronous evaluations. In *Proceedings of the World Congress on Structural and Multidisciplinary Optimization*, pages 4642–4657, 2005.

[110] Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2): 1–305, 2008.

[111] Daniel David Walker and Eric K. Ringger. Model-based document clustering with a collapsed gibbs sampler. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 704–712. ACM, 2008.

[112] Hanna M Wallach. Structured topic models for language. *Doctoral dissertation, University of Cambridge*, 2008.

[113] Hanna M Wallach, David M Mimno, and Andrew McCallum. Rethinking LDA: Why priors matter. In *NIPS*, volume 22, pages 1973–1981, 2009.

[114] Hanna M Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. Evaluation methods for topic models. In *Proceedings of the International Conference of Machine Learning*, pages 1105–1112. ACM, 2009.

[115] Xing Wei and W Bruce Croft. LDA-based document models for ad-hoc retrieval. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 178–185, 2006.

[116] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the International Conference on Machine Learning*, pages 1113–1120, 2009.

[117] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1 (6):80–83, 1945.

[118] Limin Yao, David Mimno, and Andrew McCallum. Efficient methods for topic model inference on streaming document collections. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 937–946. ACM, 2009.

[119] Ka Yee Yeung and Walter L Ruzzo. Details of the adjusted rand index and clustering algorithms, supplement to the paper "an empirical study on principal component analysis for clustering gene expression data". *Bioinformatics*, 17(9):763–774, 2001.

[120] M. Zaharia, M. Chowdhury, M.J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. In *Proceedings of the USENIX Conference on Hot Topics in Cloud Computing*, pages 10–10, 2010.

[121] Y. Zhang, Q. Gao, L. Gao, and C. Wang. iMapReduce: a distributed computing framework for iterative computation. In *IEEE International Parallel and Distributed Processing Symposium Workshops*, pages 47–68, 2011.

[122] Yanfeng Zhang, Qixin Gao, Lixin Gao, and Cuirong Wang. imapreduce: A distributed computing framework for iterative computation. *Journal of Grid Computing*, 10(1): 47–68, 2012.

[123] Yanfeng Zhang, Qixin Gao, Lixin Gao, and Cuirong Wang. Maiter: An asynchronous graph processing framework for delta-based accumulative iterative computation. *IEEE Transactions on Parallel and Distributed Systems*, 25(8):2091–2100, 2014.

[124] Ran Zheng, Genmao Yu, Hai Jin, Xuanhua Shi, and Qin Zhang. Conch: A cyclic mapreduce model for iterative applications. In *International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, 2016.

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

[126] Xiaojin Zhu, David Blei, and John Lafferty. Taglda: Bringing document structure knowledge into topic models. Technical report, Technical Report TR-1553, University of Wisconsin, 2006.