Fine-Grained Topic Models Using Anchor Words

Jeffrey Lund

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

Doctor of Philosophy

Department of Computer Science

Brigham Young University

ABSTRACT

Fine-Grained Topic Models Using Anchor Words

Jeffrey Lund
Department of Computer Science, BYU
Doctor of Philosophy

This is an abstract. It summarizes the dissertation.

Keywords: Topic modeling

ACKNOWLEDGMENTS

So long and thanks for all the fish.

Table of Contents

List of Figures

List of Tables

List of Listings

Introduction

Chapter 1

Introduction and Overview

1.1 Motivation

Note: This entire section is essentially lifted from my proposal. I need to make sure the language is appropriate for a dissertation (as opposed to a proposal).

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 email, 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.

One popular approach for understanding large collections of text is topic modeling. The goal of topic modeling is to automatically distill a large set of documents into topics. The topics, which are distributions over the vocabulary of the data, give users a general overview of the thematic contents of the entire data.

Topic models have found a wide variety of applications including document classification [12, 88], document clustering [99, 113], sentiment analysis [95], information retrieval [103], author identification [87], and more. Topic models are often used to facilitate exploratory analysis by presenting users with various visualizations of the resulting topics [25, 37]. 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 [13, 88].

This dissertation concerns itself with the problem of fine-grained topical analysis of individual documents from large collections of text. The problem of fine-grained topic modeling is two-fold: 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.

With such an analysis, we can accomplish tasks such as finding topically related documents and sentences or predicting document metadata using topical information. This problem goes beyond information retrieval or simple word correspondences 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.

As we will discuss further in Section 1.2, traditional probabilistic topic models tends to result in topic 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 [101]. While this works well for giving a glimpse of the topical content of a dataset as a whole, it is less effective when we want to examine individual documents. For example, if we used document-topic distributions learned using Latent Dirichlet Allocation [13] to find a handful of documents which are most topically related to a single target document, 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 fine-grained topics.

As an application of fine-grained topic modeling, this dissertation will focus on automatic cross-referencing (i.e., finding small sets of the most topically related documents in a large corpus). However, there are many other use cases which would benefit from fine-grained topic modeling compared to existing topic modeling techniques.

One such application is granular classification. While topic modeling features have been employed for document-level classification [88], topic models have not been successfully employed for granular classification in which individual paragraphs or sentences are labeled instead of entire documents. One use case for granular is automatic redaction of sensitive parts of documents prior to public release. While some work has been done to improve token-level topic assignment accuracy [9], the accuracy of existing topic models is such that they are only useful at a corpus level—they can reveal overall thematic trends in the data, but cannot be relied upon at a token level. We propose that fine-grained topic modeling, in which large numbers of nuanced topics are correctly assigned to individual tokens, will help mitigate the weaknesses of topic models when applied to the problem of granular document classification.

Fine-grained topic modeling could also improve exploratory analysis. With existing topic models, users are typically shown lists of related words to represent topics. Some of work has been done to help users name and interpret these word lists, but ultimately human judgement is required [57]. One way to help facilitate this judgement is to show users examples of the topic as used in the data [37]. However, this approach is difficult when token level token assignments are incorrectly assigned. As with granular classification, improved token level accuracy from fine-grained topic modeling could facilitate better understanding when topic models are used for exploratory analysis.

The majority of topic modeling literature focuses on probabilistic models such as Latent Dirichlet Allocation [13] and its many variants. While probabilistic topic modeling certainly does accomplish the task of giving a glimpse into the topical content of the entire data, for the purposes of topically analyzing specific documents from a large corpus, existing topic modeling literature has several weaknesses when it comes to fine-grained topic modeling.

More recently, new topic modeling algorithms based on non-negative matrix factorization have emerged. Ordinarily, non-negative matrix factorization is NP-hard, even when the inner dimension is small [4]. However, anchor methods rely on certain separability assumptions to solve the problem in polynomial time with provable guarantees of robustness. This separability assumption states that certain anchor words can uniquely identify a topic, meaning that those anchor words have non-zero probability in a single topic. Consequently, the anchor word occurrence patterns across the documents mirror that of the topic itself, allowing one to provably recover the topic-word distributions in polynomial time [6]. We give a more thorough review of anchor methods and related work in Section 1.2.

Unlike model based learning, the anchor algorithm scales with the size of the vocabulary rather than the size of the data. According to Heaps' Law, the number of distinct word types in documents grows exponentially slower than the total size of the text [43]. Since anchor methods scale with vocabulary size, they offer the potential to infer topic models on web scale data.

Probabilistic topic models such as Latent Dirichlet Allocation learn the topic assignments jointly with the topics themselves. We assert that learning the problems jointly makes it more difficult to learn fine-grained topics. The anchor algorithm on the other hand separates the problem of learning topics from the problem of assigning words to topics.

We argue that by extending the anchor word model to incorporate both document metadata and sets of user-specified topic words into the anchor selection process, we will be able to learn topic models with large numbers of topics which are more nuanced and specific than topics learned by probabilistic modeling.

Given large numbers of specific topics, the correct way to identify topics in documents given topic-word distributions from the anchor algorithm is also unclear. We make two contributions in this area. Topic models are typically evaluated globally using the topic-word distributions for the entire corpus without regards to the individual topic-word assignments. We will develop an automated metric for evaluating topic models locally. Using these word level evaluations, we then determine the best way to assign words to topics given a fixed set of fine-grained topics. Finally, we discuss how we validate our fine-grained topic models by applying them to the problem of automatic cross-reference generation.

1.2 Existing Topic Models

In this section, we review the most important topic modeling literature. We first give an overview of traditional probabilistic topic modeling literature, and then give a brief description of more recent work using anchor words.

1.2.1 Probabilistic Topic Modeling

One of the earliest topic models was Latent Semantic Indexing (LSI). It was first described by Deerwester et al. [29] to aid with information retrieval tasks. Later the algorithm was formally justified as a topic model [81]. 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.

A probabilistic variant of LSI, appropriately named Probabilistic Latent Semantic Indexing (or PLSI) was then developed [46]. Rather than relying on linear algebra, PLSI is a discriminative 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. [13]. 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 [93], so practitioners rely on various approximations of the posterior distribution.

Blei et al. [13] 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 approximating the posterior distribution of LDA is a collapsed Gibbs sampler [40]. 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 [72]. 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 tend to perform roughly the same when coupled with hyperparameter optimization [7]. Typically hyperparameter optimization is performed using fixed-point techniques [100]. 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 [7]. For example, Wallach et al. [101] 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 must explain many documents.

As a more concrete example, consider the problem of using topic models to automatically discover closely related passages or cross-references in a large dataset. Manually generating such annotations is typically very expensive as it requires annotators to have a deep understanding of the entire dataset so that when they read one passage, they can recall another related passage somewhere else in the data. Consequently, human annotated cross-reference datasets tend to be available for only the most well studied text, such as religious texts. Otherwise, cross-referencing tends to be limited to sparse keyword-based indexes.

Our preliminary results show that while topic-based cross-referencing has some promise, there is much room for improvement. Using a cross-reference database for the 31,102 verses of the King James version of the Holy Bible consisting of 256,726 cross-references[1], we measured the precision, recall, and F-measure of an automated cross-reference generation system based on LDA. We generate cross-references by measuring cosine similarity between document-topic distributions from LDA. We can vary the recall by adjusting a threshold on similarity to be considered a cross-reference.

The best F-measure achieved by this system was .0037, with a recall of .013 and precision of .0022. Obviously this system is in dire need of improvement before it can be useful but there is some promise. We address these short comings with fine-grained topic modeling in Chapter 7.

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 aggregately [106]. In the presence of parallel computational resources, a parallel Gibbs sampler has been developed [92]. While this

[1] Compiled using various sources including `https://www.openbible.info`, `https://www.lds.org/scriptures` and `http://www.biblestudytools.com/concordances/naves-topical-bible/`

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 [44].

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 [37]. Other popular topic visualization software include Termite [25], and a system by Chaney and Blei [21]. 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 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 [71]. Alternatively, coherence can be defined with respect to the pairwise word cooccurrences in an external dataset such as Wikipedia [77]. 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 [22]. 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 [56].

Other work on automatic evaluation of topic metric 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 [13, 88]. Topic models have also been applied to the field of information retrieval to improve ad-hoc retrieval [103]. Another task topic models have been successful at is word sense disambiguation [16].

One area in which evaluation techniques have been lacking is in token level topic assignments. As evidenced by recent work such as CopulaLDA [9], 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.

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 both document text and document metadata [12]. 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 [88]. The Multi-Aspect Sentiment model [95] and Labeled LDA [83] accomplish a similar result for sentiment and tags respectively. The Author-Topic Model generates topics which utilize authorship information during inference [87].

Other extensions of LDA seek to improve the topical structure in some way. For example the Correlated Topic Model [11] and the Pachinko Allocation [59] 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 [39] 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 [84]. 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 [33].

Integrating syntax into topic models has also proven to be useful. Both the Syntactic Topic Model [15] and HMM-LDA [41] 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 [9].

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 [48]. Rather than model 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 [26]. For the ITM, a modified Gibbs sampling scheme based on SparseLDA [106] has been proposed [47]. Alternatively, Iterated Conditional Modes [10] can be used to quickly find a locally optimal mode in the posterior in order to quickly update an ITM model [62].

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 feed 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 inference algorithms have been developed to learn such models, 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 have emerged which promises much more scalable inference. Like LSI [29], 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 $V \times K$ topic-word 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 document-topic 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 [23]. 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 [32] 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 $p(w_j \mid w_i)$: seeing word type w_j after having seen type w_i in the same document. Intuitively, $\boldsymbol{Q}$ can be thought of as $\boldsymbol{M}\boldsymbol{M}^T$, although the actual details of its construction are given by Arora et al. [6].

We then select K anchor words $\{g_1 \dots g_K\}$ 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 give 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 [43]. Consequently, at a certain dataset size, the anchor algorithm is essentially unaffected by increased scale.

In contrast, topic inference using traditional model-based approaches such as Latent Dirichlet Allocation [13] 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 [44] or Stochastic Variation Inference [45] 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 [58] propose a better anchor selection algorithm based on computing the convex hull of a t-SNE [65] 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. [79] 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

Fine-grained topic modeling combines large numbers of highly nuanced topics with correct token level topic assignments and enables topic modeling use cases, such as finding small sets of topically related documents, which depend on accurate topical analysis of individual documents. Extending anchor word topic models to incorporate document metadata and user information, and improving the quality of token level topic assignment enables fine-grained topic modeling.

1.4 Overview of Dissertation

Owing that 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 revolve around anchor selection. The anchor algorithm as described by citetanchors-practical 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 [58]. 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.

While in Chapter 2 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 is presented as a way to enabled a transparent and interactive topic-based classifier. However, by allowing metadata to influence anchor selection, enables fine-grained topic modeling by refining anchors to be more nuanced as they better reflect not only the text, but important metadata attributes.

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

First, in Chapter 5, we explore local topic model evaluation. Several models have been proposed which claim to improve the topic assignments of LDA (e.g., CopulaLDA [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 local 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 6 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 [80]. Knowing that LDA is known to have issues with local topic quality [9] 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 7 we apply what we've learned to the problem of cross cross-referencing. This problem is hard because it either requires annotators to evaluate n^2 potential references or be so familiar with the text that you might as well do the n^2 thing. LDA helps bring this down, but its still pretty bad. We do even better with fine-grained topic modeling.

TODO: Write this little blurb for real...which will require writing the chapter first to summarize

Part I

Anchor Selection

Chapter 2

Tandem Anchoring: A Multiword Anchor Approach for Interactive Topic Modeling

Published in Proceedings of the Association for Computational Linguistics 2016 [63]

TODO: I should probably add a sentence or two placing this paper in context of my dissertation.

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 [37], topic modeling is used for a wide variety of problems including document classification [88], information retrieval [103], author identification [87], and sentiment analysis [95]. However, the most compelling use of topic models is to help users understand large datasets [25].

Interactive topic modeling [49] allows non-experts to refine automatically generated topics, making topic models less of a “take it or leave it” proposition. Including humans 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 requires 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 [49], 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 $\{g_1 \ldots g_k\}$ [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 [44] or Stochastic Variation Inference [45] improves this to a single pass over the entire data. However, from Heaps' law [43] it follows that $V^2 \ll DN$ for large datasets,

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.

leading to much faster inference times for anchor methods compared to probabilistic topic modeling. Further, even if online 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 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 [58]. 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 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 [80]. 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 an 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, 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, we 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 [79]. 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. 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

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

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

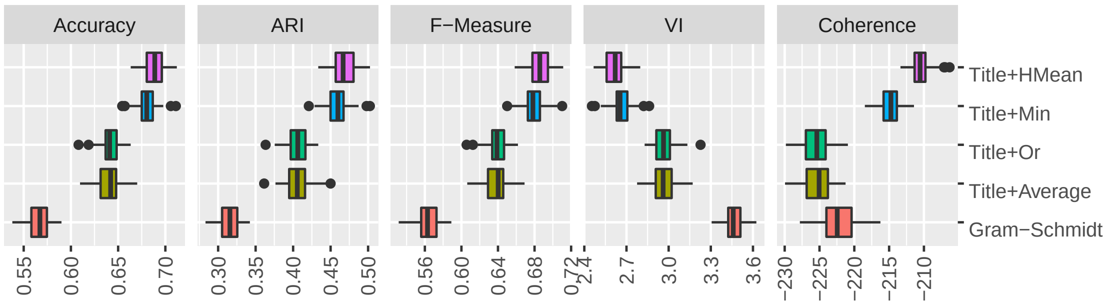


Figure 2.1: Using metadata can improve anchor-based topic models. For all metrics, the unsupervised Gram-Schmidt anchors do worse than creating anchors based on Newsgroup titles (for all metrics except VI, higher is better). For coherence, Gram-Schmidt does better than two functions for combining anchor words, but not the element-wise min or harmonic mean.

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 [13] 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.

The topics created from multiword anchor facets are more accurate than Gram-Schmidt topics (Figure 2.1). This is true regardless of the combiner function. However, harmonic mean is more accurate than the other functions.[5]

[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/`

[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 [27] and use two-tailed t-tests with Bonferroni correction to determine whether the differences between the different methods are significant.

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 [107], 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 predicted labels [69]. 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 2.1). 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. [77], 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

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

[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 [27] and use two-tailed t-tests with Bonferroni correction to determine whether the differences between the different methods are significant.

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 2.1 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.[7]

Harmonic mean beats other combiner functions because it is robust to ambiguous or irrelevant term cooccurrences 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.

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 [26] 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

[7]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 [27] and use two-tailed t-tests with Bonferroni correction to determine whether the differences between the different methods are significant.

Topic Model described by Hu and Boyd-Graber [47], and the recommended 30 iterations of sampling, the Interactive Topic Model updates with a median time of 24.8 seconds [47], 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 [23], 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 anchors 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.

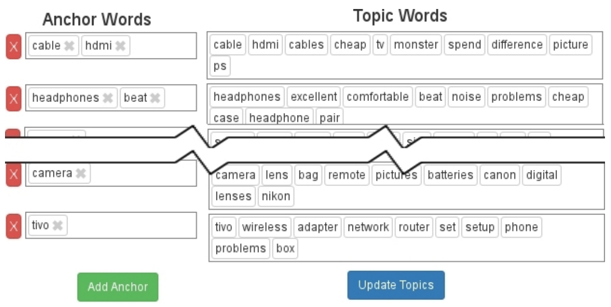

Anchor Words
Topic Words
cable hdmi
cable hdmi cables cheap tv monster spend difference picture ps
headphones beat
headphones excellent comfortable beat noise problems cheap case headphone pair
camera
camera lens bag remote pictures batteries canon digital lenses nikon
tivo
tivo wireless adapter network router set setup phone problems box
Add Anchor
Update Topics

Figure 2.2: Interface for user study with multiword anchors applied to interactive topic modeling.

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 2.2) with 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.[8] 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 a 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

[8]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.

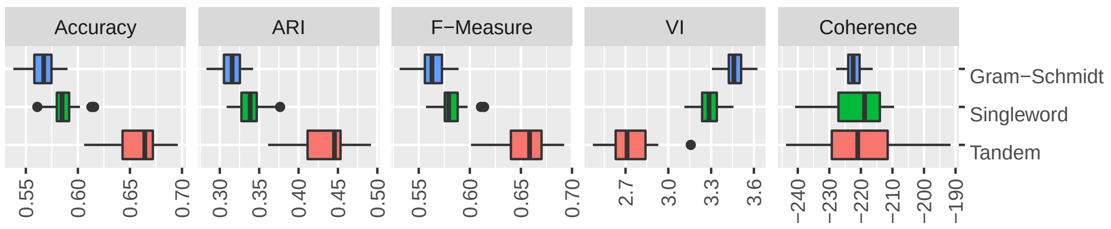


Figure 2.3: Classification accuracy and coherence using topic features gleaned from user provided multiword and single word anchors. Grahm-Schmidt anchors are provided as a baseline. For all metrics except VI, higher is better. Except for coherence, multiword anchors are best.

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.[9]

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. [49]. Basedsie 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 2.3 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 Wilcoxon's signed-rank test [105] 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 2.3). Not only is our approach more scalable than the Interactive Topic Model, but

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

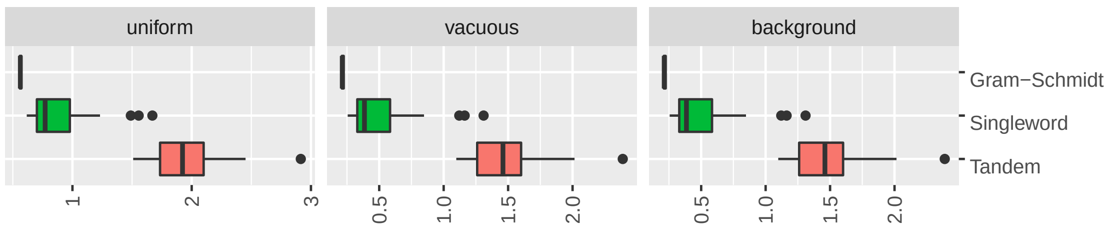


Figure 2.4: Topic significance for both single word and multiword anchors. In all cases higher is better. Multiword anchors produce topics which are more significant than single word anchors.

we also achieve higher classification accuracy than Hu et al. [49].[10] Tandem anchors also improve clustering metrics.[11]

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.[12]

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 2.4 reports our topic significance findings. For all three significance metrics, multiword anchors produce more significant topics than single word anchors.[11] 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

[10]However, the values are not strictly comparable, as Hu et al. [49] use the standard chronological test/train fold, and we use random splits.

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

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

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

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.

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.

While users find that use 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 2018 [64]

TODO: A sentences or two placing this paper within the dissertation.

Question: At some point we talked about including Supervised Anchors as part of my dissertation as well (despite only being second author on it). Does Labeled Anchors replace this chapter, or is it an additional chapter which further develops the idea?

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 [80] 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) [13] aim to automatically distill large collections of documents into topics. These topics can be used to perform document classification [88]. Furthermore, work has been done to increase the human interpretability of topics [71]. 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 [49] 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 [63].

A separate line of topic modeling research deals with supervised topic modeling, which allows document labels to influence topic inference [12]. The most recent work on supervised topic modeling is Supervised Anchors [80]. 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 [80] 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 below 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

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

number of topics (less than 200) does not noticeably increase the runtime. Furthermore, since vocabulary size tends to grow logarithmically with respect to the size of the data [43], 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 [83] 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:

$$
\begin{aligned}
\ell^* &= \underset{\ell}{argmax}\ p(\ell|\mathbf{w}) = \underset{\ell}{argmax}\ p(\ell, \mathbf{w}) && (3.3)\\
&= \underset{\ell}{argmax}\ \sum_{z_1=1}^{K} ... \sum_{z_N=1}^{K} p(\ell, \mathbf{z}, \mathbf{w}) && (3.4)\\
&= \underset{\ell}{argmax}\ p(\ell) \prod_{i=1}^{N} \sum_{z_i=1}^{K} p(z_i|\ell) p(w_i|z_i) && (3.5)\\
&= \underset{\ell}{argmax}\ \lambda_\ell \prod_{i=1}^{N} \sum_{z_i=1}^{K} \psi_{\ell,z_i} \phi z_i, w_i && (3.6)\\
&= \underset{\ell}{argmax}\ log \lambda_\ell + \sum_{i=1}^{N} log \left(\sum_{z_i=1}^{K} C_{\ell,z_i} A_{z_i,w_i} \right) && (3.7)
\end{aligned}
$$

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 [63], 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 [43], 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 [26]. 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 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. [63]. 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.

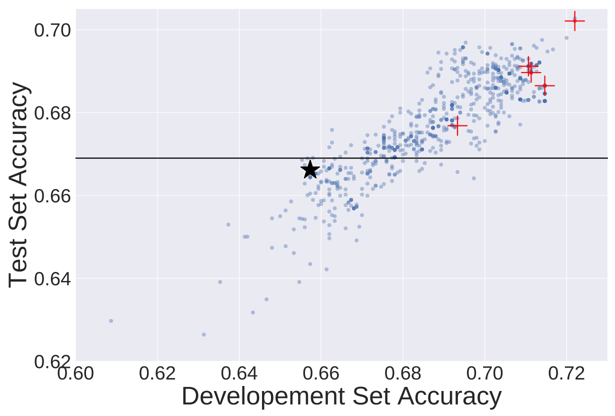


Figure 3.2: User study accuracy results comparing accuracy on the development set to the accuracy on the test set. The black horizontal line indicates the baseline accuracy from Supervised Anchors. The black star indicates the initial accuracy using Gram-Schmidt anchors with Labeled Anchors. The blue 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. Most of the multiword anchors users used were short 2-3 word phrases which did not have an obvious single word counterpart.

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.

Chapter 4

Free Classifier Extras

- Breakdown of training times
- Overwatched for time savings
- Follow up study showing that number of edits matters

Part II

Token Assignment

Chapter 5

Metrics

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

Abstract

Global topic evaluation is nice. Local topic evaluation would be nice. We show you how. You should do both.

5.1 Introduction

Topic models such as Latent Dirichlet Allocation (or LDA) [13] 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 huge variety of different models ranging from models which learn and predict document labels [12, 88] or document sentiment [95], models which incorporate syntax and other linguistic features [15, 41, 114], models which interactively incorporate outside domain knowledge [3, 49], and a variety of other models which perform specialized analysis on specific data [20]. Topic modeling is useful not only for automated tasks, but also for facilitating manual exploratory analysis [25, 37].

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

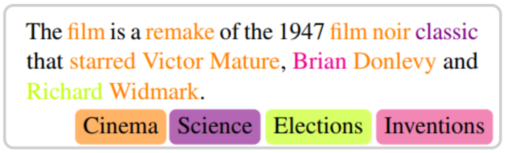

The film is a remake of the 1947 film noir classic that starred Victor Mature, Brian Donlevy and Richard Widmark.

Figure 5.1: LDA on a sentence from a Wikipedia document. Notice that even noun-phrases are split in a way which is bewildering to users.

is possible (e.g., document classification). In most other cases, practitioners rely on automated metrics such as topic coherence [77] in order to compare topic model quality.

We review coherence and other existing metrics for topic model evaluation in Section 5.2. 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, topic models typically attribute each individual token to a specific topic. These token-level topic assignments are often used as features for downstream tasks, or are used to help understand the topical content of individual documents. Unfortunately the quality of the local topic assignments can be bewildering to users.

TODO: Get my own example and replace this cinema thing

For example, Figure 5.1 shows typical topic assignments using LDA. Arguably, the entire sentence should be assigned to the 'Cinema' topic, but parts of the sentence are assigned to other topics, possibly because other sentences in the same document are concerned with those topics. Most troubling is the fact even noun-phrases, which presumably should be assigned to the same topic, are split across different topics.

TODO: Add a paragraph explaining why this matters. There should be some application we enable, but I don't want to claim things I can't prove.

The problem of improving local topic assignments has received some attention from the modeling-side in topic modeling literature. For example, HMM-LDA [41] integrates syntax and topics by allowing words to generated from a special syntax specific topic. TagLDA [114] 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. [8] proposed SentenceLDA which assigns each sentence to a single topic. Alternatively, CopulaLDA [9] 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 typically evaluated with respect to global topic quality instead. 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 ascertain which topic models are indeed good solutions for topically examining individual documents. Following the example of previous work on global topic evaluation [77], in Section 5.3 we first propose a variety of automated metrics designed to perform this evaluation. Then, in Section 5.4 we propose a user study designed to elicit human evaluations of local topic quality. We detail the results of this study and correlate those results with our proposed metrics in Section 5.5. Finally, we conclude with a discussion of our findings in Section 5.6.

5.2 Global Evaluation

Early topic models such as LDA were typically evaluated using held-out likelihood or perplexity [13]. [102] gives details on how to estimating 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 [22]. These judgements 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. This task operates under the assumption that if a topic is semantically coherence, then the intruder should be easy to identify.

As alternative to held-out perplexity for evaluating topic coherence, several methodologies have been introduced. Newman et al. [77] proposed evaluating topic models by aggregating pairwise PMI scores across the top n most likely terms in a topic distribution, while Mimno et al. [71] proposed a metric using conditional probability of the top n words. The PMI scores or conditional probabilities can be computed with respect to the modeled data, but more frequently are computed using some large reference corpus such as Wikipedia. This style of topic modeling evaluation, This metric, colloquially referred to simply as coherence, is currently the most popular form of automated topic model evaluation. Note that coherence is a measure of global topic quality, since it considers only the global topic-word distributions, without regard to local topic quality or token level topic assignments.

Coherence has been well studied. For example, through certain types of regularization, we can improve topic coherence [78]. Newman et al. [77] 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 [56].

Since topics are typically summarized by their top n most probable words, for the purpose of exploratory analysis, topic coherence is an important consideration. 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. Consequently, AlSumait et al. [1] developed metrics for evaluating topic significance. Originally, this work was intended to rank individual topics by measuring the distance each individual topic distribution from some background or "junk" distribution. However, these significance metrics have also been used to characterize entire models by measuring the average distance over all topics in a model from a background distribution [63].

These background distributions include the uniform distribution over the corpus vocabulary and the empirical distribution of words in the corpus. Alternatively, instead of measuring the topic-word distributions, we can measure the distance of distribution of topics across documents from the uniform distribution over documents.

Like coherence, topic significance is a global measure of topic quality since it considered the topic distributions without regard to local topic assignments. Because significance considered the entire topic distribution, and not just the top n words of a topic, it can differ from coherence in important ways. For example, Lund et al. [63] found that when topics were used as features for document classification, models with similar coherence scores might perform differently on downstream classification accuracy depending on the significance scores since highly significant topics were more likely to be useful features than an equally coherence, but less significant topic feature.

5.3 Proposed Metrics

Even recent models such as CopulaLDA [9] which claim to improve the quality of token level topic assignments have only been evaluated using global topic metrics. We aim to develop an automated methodology for evaluating local topic model quality. We follow the pattern used by Newman et al. [77] to develop coherence and will propose a variety of potential metrics. In subsequent sections, we will correlate these automated metrics with human evaluations in order to determine which automated metric yields the most accurate estimate of local topic quality.

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

$$1 - \frac{1}{n-1}\sum_{i=1}^{n-1} \delta(z_i, z_{i+1}) \tag{5.1}$$

Essentially what this metric measures is the percent of times a topic switch occurs relative to the number of times it could have switched. We subtract this percent from one simply so that we can have the metric measure local topic consistency rather than inconsistency.

Topic Switch VI Topic switch percent penalizes any topic switch without regards to how related the topics are. We utilize variation of information, which measures the amount of information lost in changing from one partition to another [69]. Once again using z_i as the topic assignment of the ith word in a corpus with n words, we consider two partitions $S = \{z_1, z_2, z_3, ..., z_{n-1}\}$ and $T = \{z_2, z_3, z_4, ..., z_n\}$. Variation of information is defined as

$$H(S) + H(T) - 2I(S, Y) \tag{5.2}$$

where $H(\cdot)$ is entropy and $I(S, Y)$ is the mutual information between S and T. Similar to topic switch percent, this measure penalizes topic switches. However, models which only switch between highly related topics will be penalized less than models which switch to topics at random.

Average Rank Even when evaluating local topic quality, the most common way of representing topics is as a set of related words, namely the most probable words in the topic-word distributions. Consequently, human evaluations of local topic quality will still be influenced by the ranking of a word in its assigned topic. Leveraging this intuition, where $rank(w_i, z_i)$ is the rank of word i in topic i, we use the following:

$$\frac{1}{n}\sum_{i=1}^{n} rank(w_i, z_i) \tag{5.3}$$

Topic-Word Divergence

Window Probabilities

TODO: Describe worddiv
TODO: Describe windowp

5.4 Study Design

We follow the general design philosophy employed by Newman et al. [77] 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 with a task designed to elicit human evaluation of

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

Table 5.1: Statistics on datasets used in user study and metric evaluation.

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, and then crowdsourcing task.

5.4.1 Models

We choose three different dataset from 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. We applied standard tokenization and stopword removal. Additionally, we remove any token which does not appear in at least 100 documents. Statistics for these three datasets can be found in Table 5.1.

On each of these datasets, we train three types of topic models. As a baseline, we train Latent Dirichlet Allocation [13] on each of the three datasets. CopulaLDA [9] reportedly improves local topic quality, so we also employ this model. 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. [80] 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. 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

[1]Unfortunately, CopulaLDA did not scale beyond 100 topics.

of more significant topics, allowing us to observe user preferences on a variety of models. Additionally, we trained multiple instances of each model to account for non-determinism.

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

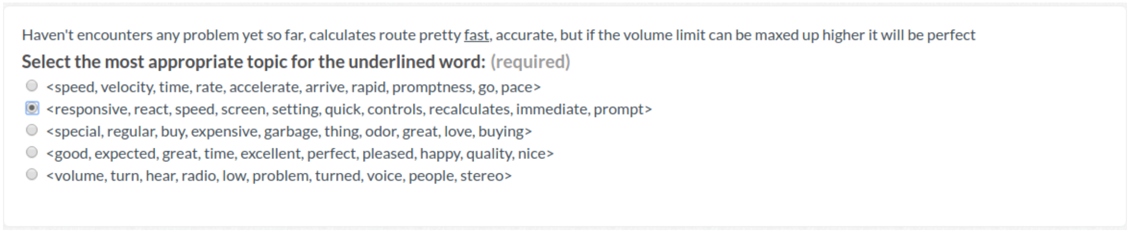
5.4.2 Crowdsourcing Task

Note: We do *not* need an IRB since we are asking questions about models. We obtain no information about a living individual. We have a quasi-contractual relationship with the crowd source workers to obtain the information about those models.

Our goal in designing a crowdsourcing task is to get human annotators to evaluate the quality of token level topic assignments. 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 crowdsourced evaluations of topic coherence relied on the word intrusion task instead of asking annotators to directly rate topic coherence [22].

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 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 more accurate topic model. 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, but an accurate model should be able to distinguish between topics in the same document, and the topic for a given token (although as seen in Figure 5.1, models such as LDA can struggle with this). Finally, we include a randomly selected intruder topic as a fifth option. This fifth option is include to help distinguish between the case when the user sees equally reasonable topics for

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


Haven't encounters any problem yet so far, calculates route pretty fast, accurate, but if the volume limit can be maxed up higher it will be perfect

Select the most appropriate topic for the underlined word: (required)

- <speed, velocity, time, rate, accelerate, arrive, rapid, promptness, go, pace>
- [✓ <responsive, react, speed, screen, setting, quick, controls, recalculates, immediate, prompt>]
- <special, regular, buy, expensive, garbage, thing, odor, great, love, buying>
- <good, expected, great, time, excellent, perfect, pleased, happy, quality, nice>
- <volume, turn, hear, radio, low, problem, turned, voice, people, stereo>

Figure 5.2: Example of crowdsourced task. Users are asked to select the topic which best explains the underlined token.

the underlined token, and no reasonable options for the underlined token. Figure 5.2 shows an example of this task shown to annotators.

For each trained model (i.e., for each model type, dataset, and topic cardinality), we randomly selected 1000 words to annotate. For each selected word, we obtain 5 judgements. We aggregate the 5 judgements by using the contributor response with the highest confidence, with agreement weighted by contributor trust.

We deployed this task on a popular crowdsourcing website[3] and paid contributors 0.12 USD per page, with 10 annotations per page. For quality control on this task, we each page contained one test question. The test questions in our initial piloted studies were questions we hand-selected for their obvious nature. In the final study we ran to obtain the results in Section 5.5 we also selected a number of questions with both high annotator confidence and perfect agreement to augment our bank of test questions. We required 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 imposed no other constraints on contributors.

5.5 Results

In this section, we report the findings of our user study and correlate the human judgements with our automated metrics.

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

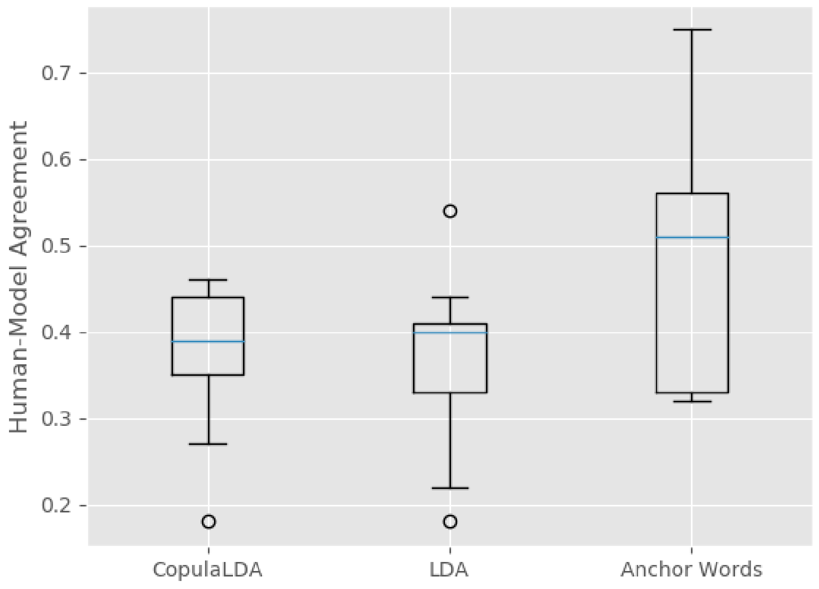


Figure 5.3: Plot showing human agreement with each model type. CopulaLDA performs slightly worse than LDA. Humans preferred topic assignments from Anchor Words by a wide margin.

We first measure inter-annotator agreement using Krippendorff's alpha with a nominal level of measurement [53]. Over all the judgements we obtained, we computed a value of $\alpha = .44$. While this is nowhere near perfect agreement, this does indicate a moderate level of agreement.

TODO: Find citation about agreement in crowdsourcing.

Figure 5.3 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.

Question: Are these claims too strong or rude? I don't want to bash the CopulaLDA guys *too* much...

Equally surprising is the fact that users agreed with Anchor Words more often than LDA by a wide margin, indicating that Anchor Words achieves superior token level topic assignments. In terms of global topic quality, Anchor Words is roughly similar to LDA [6].

	Metric	Amazon	Newsgroups	New York Times
local	switchp	0.9077	0.8737	0.7022
	switchv	0.8485	0.8181	0.6977
	avgrank	0.5103	0.5089	0.4473
	worddiv	0.3112	0.2197	0.0836
	windowp	0.4884	0.3024	0.1127
global	coherence	0.4907	0.4463	0.3799
	sigwuni	0.6310	0.4839	0.4935
	sigwvac	0.6960	0.6081	0.6063
	sigdback	0.0655	0.1121	0.0798

Table 5.2: 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.

Further investigation into this phenomenon is warranted, but we believe that Anchor Words can achieve superior local topic quality because the problem of recovering the global topic-word distributions is separate from the problem of producing local topic assignments, making both tasks easier.

Question: Is this supposition too strong? I really believe it, but until I write the next chapter, I can't really prove it.

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 5.2, we report the coefficient of determination (r^2) for each metric and dataset.

Question: Are there r^2 for coherence we could compare to just to see how strong our r^2 is in comparison?

While switchp is the simplest of our proposed metrics, across all datasets, switchp most closely approximate human judgements 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 topicv, even switching between related topics seems to contradict human judgements of local topic quality.

TODO: Figure out what to say about why Amazon matches humans more than New York Times.

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.

5.6 Discussion

Question: Is this tone okay?

Our contributions are thus: we have proposed a new crowdsourcing task, namely topic-word matching, to elicit human judgements on the quality of token level topic assignments; we have obtained human judgements on this task on several models and several datasets; finally, we have developed an automated metric which correlates well with these human judgements of local topic quality.

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-based metrics 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 do not advocate that topic consistency be adopted 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 global topic quality (i.e., coherence and significance) 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.

Question: Is there anything else we should say here?

TODO: Added something about how cool it is that the consistency metric matches what we intuitively would have guessed, despite the fact that we didn't directly ask.

Chapter 6

Token Level Topic Assignments

Abstract

This is an abstract.

6.1 Introduction

In Chapter 5, we established topic consistency (measured by the percent of words which switch topics from the preceding word) as an effective proxy for automated 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 [32], 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 numerically unstable matrix inversion [4]. Consequently, practitioners utilizing the Anchor Words algorithm typically infer per word topic assignments using Latent Dirichlet Allocation [13] with fixed topics[1] learned with the anchor algorithm. This inference is typically performed using mean field variational inference [80], since it has been shown to perform well for Latent Dirichlet Allocation in general [7].

[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 [42].

However, other inference techniques have been used in conjunction with Latent Dirichlet Allocation. Since the problem of exact inference inference is NP-Hard [93], exact techniques such as expectation propagation [73] or belief propagation [82] have mostly been ruled out due to scalability issues. Instead, approximate inference algorithms such as expectation maximization [31] and Gibbs sampling [40] 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 [100], variational inference performs as well as Gibbs sampling while being much more scalable [7].

More recent work has highlighted the potential usefulness of iterated conditional modes as an inference technique for topic models [62]. While iterated conditional modes has been known for some time [10], 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 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.

6.2 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 [97] 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:

$$q(\theta) = \prod_i q_i(\theta)i) \tag{6.1}$$

Using the mean field assumption, we can perform updates similar in spirit 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 [13]. For our experiments, we utilize online stochastic variational updates [44] 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 [40]. A Markov chain is obtained by iteratively sampling the topic assignment for each word according to the full conditional

$$p(z_{di}|z_{\neg di}, w) \propto (n_{d,z_i,\cdot} + \alpha) \frac{n_{\cdot,z_i,w_i} + \beta}{n_{\cdot,\cdot,w_i} + V\beta} \tag{6.2}$$

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.

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

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

Iterated Conditional Modes

Iterated Conditional Modes 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 simple means applying the *argmax* to Equation 6.2. 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 quite some time [10], it has recently been given attention in topic modeling literature for the purpose of interactive topic modeling [62]. While Iterated Conditional Modes is fast enough even for interactive topic models, because fundamentally the algorithm is coordinate-wise ascent, 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 using a uniform distribution over topics. Uniform 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 taking the best of just 10 random restarts, we were able to significantly boost performance.

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

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 gives the most mass to the individual word. Because of polysemy, and the fact that Equation 6.2 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.

Per-Sentence Initialization?

Some other model?

Added regularization?

6.3 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 Word algorithm: one for standard coarse-grained topic modeling, and another with a larger number of topics for fine-grained topic modeling.

For both set 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 be able to meaningfully choose 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 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

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

coarse granularity. On the other hand, if we used 1000 topics, each topic must only explain an average of about 19 documents, which is much more fine grained[4] Table 6.1 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 will then employ each of the proposed assignment techniques discussed in Section 6.2. Since the topic-word distributions are fixed, global evaluations such as coherence do not useful for distinguishing between token assignment strategies. However, as discussed in Chapter 5 there are several ways we can evaluate the local topic quality. Most importantly, we will measure the local topic consistency by computing the percent of words which share a topic with the previous word in a document. We can also measure topic significance by measuring the distance of the uniform distribution over topics with the distribution of topics across documents [1]. Finally, we can use a downstream task as a proxy, namely document classification, as a proxy for local topic quality. To this end, 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. Unfortunately,

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

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 6.2: Topic consistency results. Higher is better. Coarse topics results is more consistent models. However, regardless of topic granularity, ICM+Word Init yields the most locally consistent topic assignments.

for our New York Times data, we have no predictable metadata, so that dataset will only be evaluated with respect to consistency and significance.

TODO: Need more detail throughout. For example, what classifier are we using? What train/test split is happening? What features?

6.4 Results

Table 6.2 reports the topic consistency results as measured by the percent 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 on the local topic consistency. Most notably, Gibbs sampling yields very inconsistent results, while Iterated Conditional Modes more locally consistent. Perhaps most interestingly is the fact that while Iterated Conditional Modes yields the most consistent result, no one initialization strategy dominated the others, meaning that maximizing the probability of the data does not necessarily correlate with a more locally consistent topic model.

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 6.3: Topic significance as measured by divergence from the background document-topic distribution. Higher is better. Iterated Conditional Modes yields the best performance, although depending on the dataset different initialization strategies may work better than others.

The higher consistency of Iterated Conditional Modes is explained by the topic significance results shown in Table 6.3. 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 that Iterated Conditional Modes outperforms both Gibbs sampling and Variational Inference. Since each document is more 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 us to get a more nuanced view of individual documents. As an example, consider two sets of topics learned from the 20 Newsgroups dataset which deal with religious issues shown in Table 6.5. 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

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

Table 6.4: Classification accuracy results. When paired with a good assignment strategy, fine-grained topics improves classification accuracy over standard coarse-grained topics.

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 6.5: 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, while for fine grained, only a small sample of the religious topics are shown. Fine-grained topics are much more nuanced and detailed.

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 handful of them are shown in Table 6.5). 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 so-called the problem of evil found in both 'alt.atheism' and 'soc.religion.christian'.

Undoubtedly, 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 exhibits much lower topic significance, meaning that these nuanced topics are much more evenly spread out across documents, making it much harder to find specific documents or passages which are closely relate to a fine-grained topic. Remembering that in this case, topic significance is measured using Kullback-Leibler divergence of the document-topic distribution from the uniform distribution over topics, we can interpret the significance numbers as information gain with respect to the uniform distribution. We can therefore say that a good topic assignment strategy such as Iterated Conditional Modes with Per-Word Initialization overs 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.

.

TODO: Example topics so we can show why significance matters

Question: It might be good to include an appendix list all the topics so they can examine them in full, or even just give a sample for qualitative eval.

Another place where we can see the effects of topic significance is with respect to downstream classification accuracy. Table 6.4 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 accuracy suggest that not only does this strategy offer better topic significance than many of the alternatives, but the assignments it makes are more accurate than the competing assignment strategies.

6.5 Conclusion

This chapter has provided two important contributions. First, we have demonstrated 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. Surprisingly, Gibbs sampling is a poor assignment strategy despite its popularity with Latent Dirichlet Allocation.

Second, we have demonstrated that when combined with an effective topic assignment strategy, we can dramatically 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.

Part III

Application

Chapter 7

Xref

7.1 Introduction

Cross references, or references to other parts of a text which elaborate upon or clarify a particular passage of text, can be an invaluable tool for deep understanding of a text. They 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 information.

However, the process of creating such a resource can be expensive and time consuming. For example, the creation of the LDS Standard Edition of the Holy Bible, which added numerous cross references and topic based categories, took hundreds of volunteers thousands of hours over seven years to produce [2]. This processed involved the collection of manually curating more than 19,900 entries from volunteers, and then editing and refining those references with a small committee of experts.

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 remember. Alternatively, we can make individual annotations easier by showing annotators two passages and asking

Part IV

Parallelization

Chapter 8

Parallelization of Anchor Algorithm

In this chapter, we briefly justify the next chapter being included.

Chapter 9

Mrs: high performance MapReduce for iterative and asynchronous algorithms in python.

Published in Workshop on Python for High-Performance and Scientific Computing 2016 [61]

Abstract

Mrs [67] 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).

9.1 Introduction

Mrs [67] 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 9.3.1). 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 9.3.2). Third, we present a generator-callback model for submitting operations for concurrent and asynchronous evaluation (Section 9.3.3). 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 9.4 which efficiently supports algorithms such as Particle Swarm Optimization (PSO) [68] 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 9.2.

Finally, we demonstrate the application of these techniques in Mrs [67]. Section 9.5 evaluates the performance of Mrs with and without these features, using PSO [68] and expectation maximization (EM) [24] 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%.

9.2 Related Work

MapReduce [28] 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 [19, 50, 76, 108]. Frameworks like Maiter [111] and GraphLab [60] 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 [112] and Twister [34] 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 [18, 74, 91]. 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 [18, 74, 86, 112]. Another technique implemented in iMapReduce [109, 110] 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 [35] primary modification to Hadoop was the addition of asynchronous scheduling, but several frameworks have since implemented some sort of asynchronicity into their designs [91, 109–111].

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 9.3 and 9.4, as well as the generator-callback model to handle task scheduling and completion.

9.3 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 9.3.1 shows a principled approach for limiting the frequency of checkpoints to distributed storage. Section 9.3.2 describes a reduce-map operation for agglomerating reduce and map tasks. Section 9.3.3 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 9.5.

9.3.1 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 [75]. 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 [34], 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.

9.3.2 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 [35, 109]. 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 9.1 shows the dependencies between

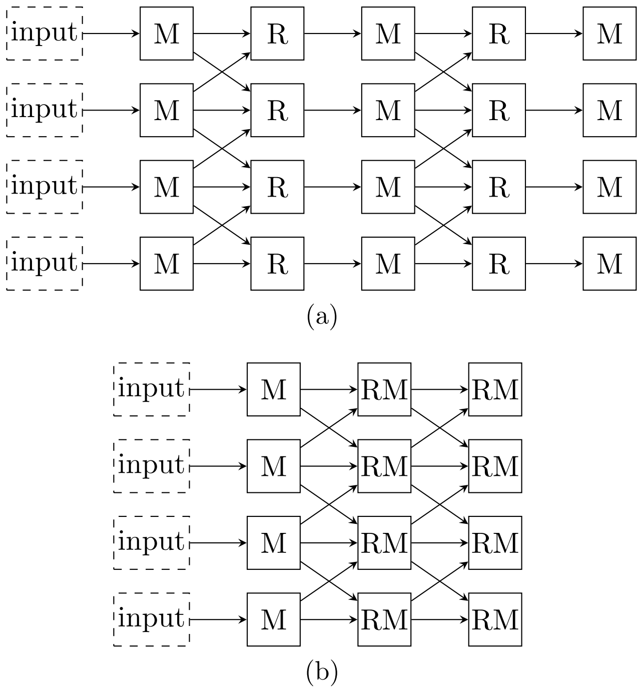
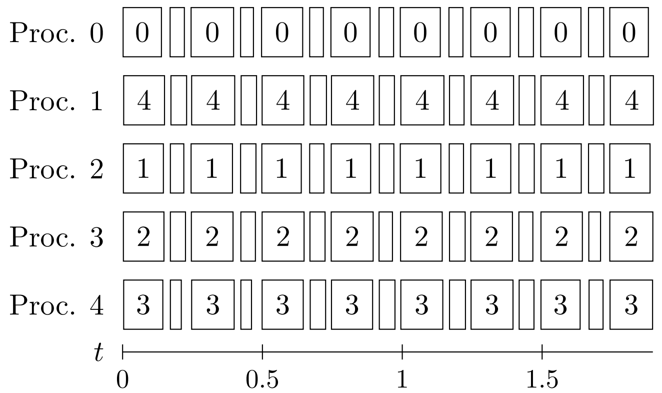


Figure 9.1: Task dependencies of a typical iterative MapReduce program with (a) standard map (M) and reduce (R) operations, contrasted with (b) combined reduce-map (RM) operations.

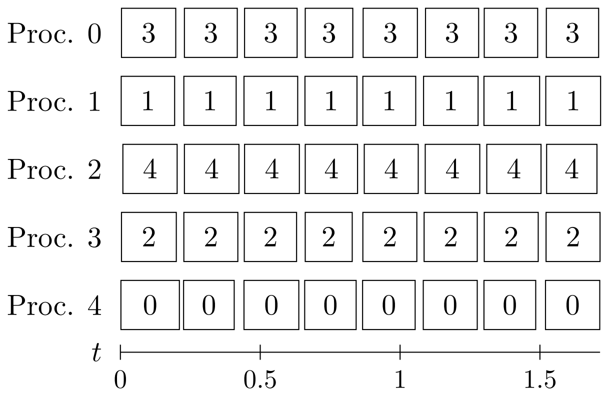
tasks with separate reduce and map tasks (Figure 9.1a) and with combined reduce-map tasks (Figure 9.1b).

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 9.2 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 9.2: Actual task execution traces generated from a sample application (particle swarm optimization, see Section 9.5.1 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.

9.3.3 Iterative Programming Model

The standard MapReduce model defines a single map phase followed by a single reduce phase [28], 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 [108]. 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 [76], 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. Program 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. Program 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

Algorithm 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
```

Algorithm 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)
```

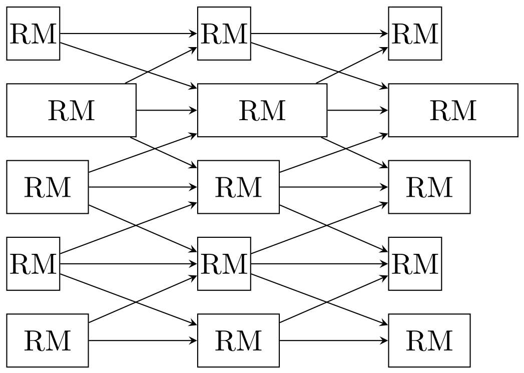
to reduce overhead. These convergence checks are performed concurrently with subsequent iterations and could be submitted as datasets if they represent significant computation. The simple generator-callback structure makes it easy to specify computation that varies from iteration to iteration and to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

9.4 Asynchronous MapReduce Programming Model

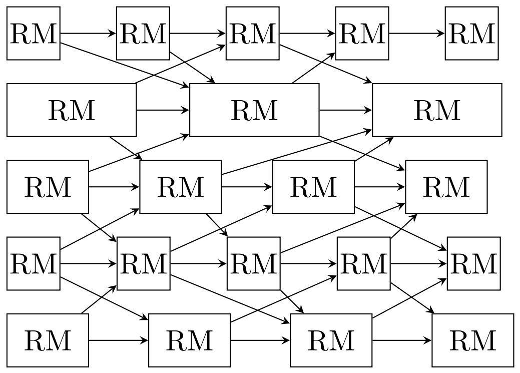
Iterative MapReduce can serve as a simple message passing framework. A map task serves to update an object, emit it, and emit messages to other objects. Between map tasks and reduce tasks is an implicit barrier for communication to complete, and a reduce task aggregates messages and emits the object, 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 9.3 illustrates task dependencies in an iterative program with heterogeneous task execution times.

In synchronous MapReduce (Figure 9.3a), the barrier betweeen iterations leaves the faster processors idle, but in asynchronous MapReduce (Figure 9.3b), 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 9.3: 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`.

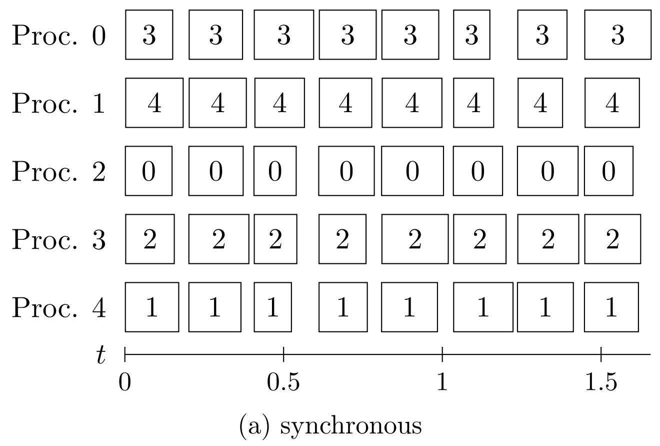
The `async_start` parameter is a boolean indicating whether a dataset can start asynchronously while some tasks in its input are still running. The `blocking_ratio` parameter determines the minimum fraction of tasks that must be completed before any child dataset can start asynchronously and defaults 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 9.5.1, is an example of a simple iterative message passing algorithm that is naturally expressed in MapReduce [68]. 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 [52, 96]. The fully distributed variant of asynchronous parallel PSO makes its message passing nature particularly clear [90].

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 9.4 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



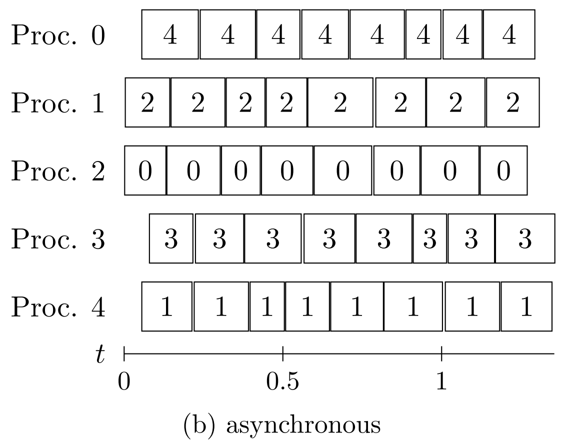


Figure 9.4: Actual task execution traces for PSO with synchronous and Asynchronous MapReduce. The horizontal axis is measured in seconds.

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.

9.5 Experimental Results

Although the approaches described in this paper are applicable to any MapReduce implementation, we evaluate their effects using the Mrs [67] 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 [38], for which we use typical serial implementations.

9.5.1 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 9.5.1 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 9.5.1 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 [17, 51]. 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 [68]. 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 [85, 89].

Program 3 is an implementation of PSO using a generator-callback API as in Program 2 from Section 9.3.3.

We find significant performance improvements for PSO in MapReduce. We use PSO with subswarms of 5 particles applied to the 250 dimensional Rosenbrock function [94]. 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

Algorithm 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
```

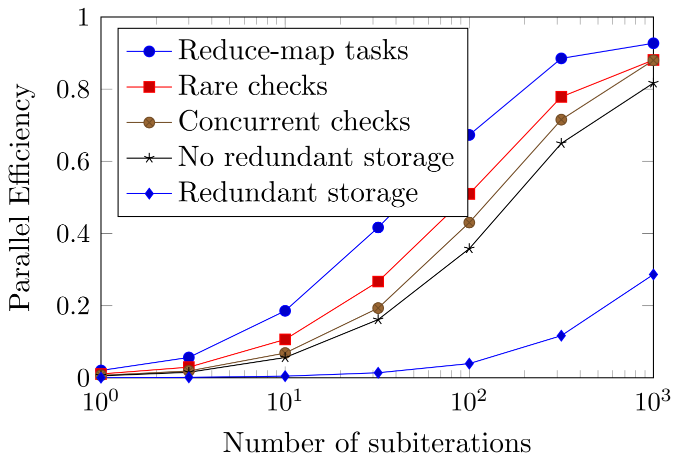


Figure 9.5: Parallel Efficiency (per iteration) of PSO in MapReduce with a sequence of cumulative optimizations. The x-axis represents the problem size (the number of subiterations in each map task). “Redundant storage” represents the baseline performance, with all data stored to a redundant filesystem 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 9.5 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 [30]. Specifically, we apply the algorithm to a mixture of multinomials model in the context of clustering text documents [70, 98]. 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 [54, 66]. 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 [55]. After preprocessing, the dataset had a vocabulary size of approximately 80,000 unique words. As a final step, we applied random

Table 9.1: 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 [36, 104], but other more principled dimensionality reductions such as latent dirichlet allocation [14] could also be used to reduce the feature set size.

Table 9.1 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.

9.5.2 Asynchronous MapReduce

The asynchronous programming model of Section 9.4 allows asynchronous parallel PSO [52, 96] 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 distributed

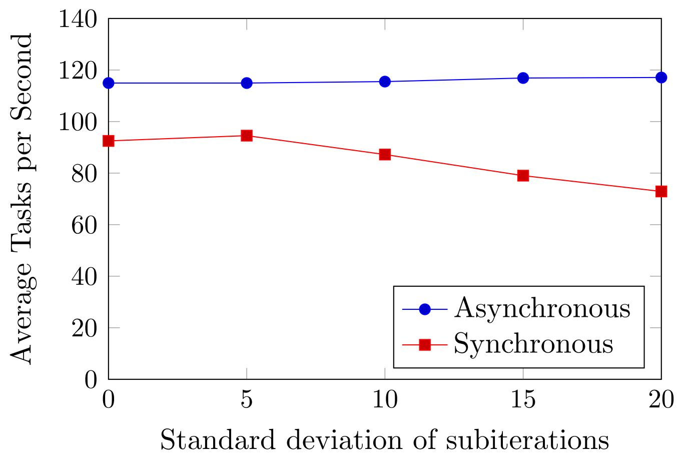


Figure 9.6: The average throughput (in tasks per second) for synchronous and asynchronous PSO. The number of subiterations per map task vary, with an average of 50 and a standard deviation ranging from 0 to 20. Throughput of the asynchronous implementation is unaffected by task variance and is better even when there is no variance.

environments [90]. 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 9.6 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.

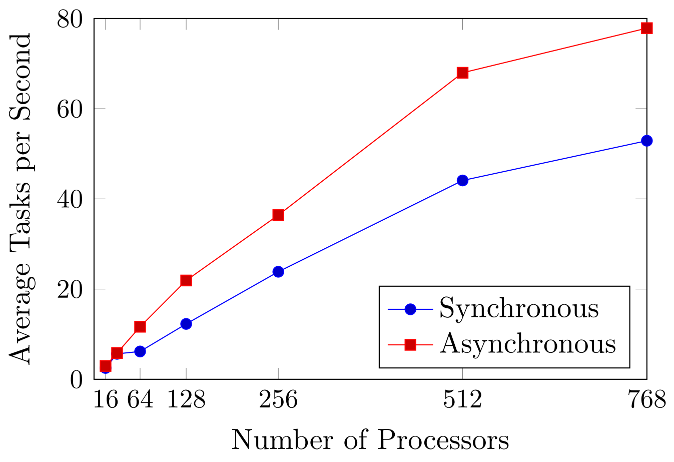


Figure 9.7: The average throughput (in tasks per second) for synchronous and asynchronous PSO with respect to the number of processors. The number of subswarms is equal to the number of processors, and the number of subiterations is 1000.

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 9.7 explores this phenomenon and shows that the effect increases with the number of processors.

9.6 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.

Conclusion

Chapter 10

Conclusions

This is where I'll pretend it was all worth it.

References

[1] Loulwah AlSumait, Daniel Barbará, James Gentle, and Carlotta Domeniconi. Topic significance ranking of LDA generative models. In *Proceedings of European Conference of Machine Learning*, 2009.

[2] Lavina Fielding Anderson. Church publishes first LDS edition of the bible. *Ensign*, 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*, 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*, 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*, 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*, 2013.

[7] 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.

[8] 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*, 2016.

[9] 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.

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

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

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

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

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

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

[16] 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*, 2007.

[17] D. Bratton and J. Kennedy. Defining a standard for particle swarm optimization. In *Proc. IEEE Swarm Intelligence Symposium*, 2007.

[18] Y. Bu, B. Howe, M. Balazinska, and M. Ernst. HaLoop: Efficient iterative data processing on large clusters. *Proc. VLDB Endowment*, 3(1-2), 2010.

[19] C. Chambers, A. Raniwala, F. Perry, S. Adams, R.R. Henry, R. Bradshaw, and N. Weizenbaum. FlumeJava: Easy, efficient data-parallel pipelines. In *ACM Sigplan Notices*, 2010.

[20] Allison Chaney, Hanna Wallach, Matthew Connelly, and David Blei. Detecting and characterizing events. In *Proceedings of Empirical Methods in Natural Language Processing*, 2016.

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

[22] 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.

[23] 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.

[24] 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 *Proc. Advances in Neural Information Processing Systems*, 2007.

[25] 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*, 2012.

[26] 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.

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

[28] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *Proc. Operating System Design and Implementation*, 2004.

[29] 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.

[30] 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.

[31] 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.

[32] 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*, page None, 2003.

[33] 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.

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

[35] E. Elnikety, T. Elsayed, and H.E. Ramadan. iHadoop: Asynchronous iterations for mapreduce. In *Proc. IEEE Cloud Computing Technology and Science*, 2011.

[36] Kuzman Ganchev and Mark Dredze. Small statistical models by random feature mixing. In *Proc. Workshop on Mobile NLP at ACL*, 1998.

[37] 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.

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

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

[40] 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.

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

[42] Joshua Aaron Hansen. Probabilistic explicit topic modeling. 2013.

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

[44] Matthew Hoffman, Francis R Bach, and David M Blei. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, 2010.

[45] 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.

[46] 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.

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

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

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

[50] 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.

[51] James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *Proc. International Conference on Neural Networks IV*, 1995.

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

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

[54] 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*, 2009.

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

[56] 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*. Association for Computational Linguistics, 2016.

[57] Jey Han Lau, Karl Grieser, David Newman, and Timothy Baldwin. Automatic labelling of topic models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1536–1545. Association for Computational Linguistics, 2011.

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

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] 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*, 2017.

[64] 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*, 2018.

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

[66] 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*, 2009.

[67] A. McNabb, J. Lund, and K. Seppi. Mrs: MapReduce for scientific computing in Python. In *Proc. Python for High Performance and Scientific Computing*, 2012.

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

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

[70] Marina Meila and David Heckerman. An experimental comparison of model-based clustering methods. *Machine Learning*, 2001.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

[75] E. Morenoff and J.B. McLean. Application of level changing to a multilevel storage organization. *Communications of the ACM*, 10(3):149–154, 1967.

[76] 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 *Proc. Network Systems Design and Implementation*, 2011.

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

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

[79] 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*, 2014.

[80] 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*, 2015.

[81] 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.

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

[83] 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.

[84] 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.

[85] Juan Romero and Carlos Cotta. Optimization by island-structured decentralized particle swarms. In *Proc. Fuzzy Days: Computational Intelligence, Theory and Applications*, 2005.

[86] 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.

[87] 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*, 2004.

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

[89] 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), 2004.

[90] Ian Scriven, David Ireland, Andrew Lewis, Sanaz Mostaghim, and Jürgen Branke. Asynchronous multiple objective particle swarm optimisation in unreliable distributed environments. In *Proc. IEEE Congress on Evolutionary Computation*, 2008.

[91] 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.

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

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

[94] 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.

[95] 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*, 2008.

[96] Gerhard Venter and Jaroslaw Sobieszczanski-Sobieski. A parallel particle swarm optimization algorithm accelerated by asynchronous evaluations. In *Proc. World Congress on Structural and Multidisciplinary Optimization*, 2005.

[97] 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.

[98] Daniel Walker and Eric Ringger. Model-based document clustering with a collapsed gibbs sampler. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

[99] 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.

[100] Hanna M Wallach. Structured topic models for language. *Unpublished doctoral dissertation, Univ. of Cambridge*, 2008.

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

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

[103] 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*, 2006.

[104] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proc. International Conference on Machine Learning*, 2009.

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

[106] 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.

[107] 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.

[108] M. Zaharia, M. Chowdhury, M.J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. In *Proc. USENIX Conference on Hot Topics in Cloud Computing*, 2010.

[109] 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*, 2011.

[110] 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.

[111] 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.

[112] 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.

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

[114] 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.