Chapter 1

Introduction

Topic modeling is an unsupervised machine learning technique that analyzes a text corpus to determine topic identification and distribution across documents. It is similar to soft clustering where each document can belong to multiple topics in varying proportions. The need for topic modeling is addressed to reduce the manual effort required to explore, identify, and organize text corpora. This machine learning algorithm addresses the time gap by automating the process of organizing by extracting meaningful patterns and themes from large collections of text. The vast applications of topic modeling, include customer feedback analysis[1], typically seen in Amazon product reviews, where it helps understand consumer sentiments. Additionally, in email filtering, institutions and businesses that are quite used to the high volume inflow of emails would benefit from topic modeling by making use of efficient categorization of emails to mark them as "Inquiry", "Complaint", "Suggestion", etc. Furthermore, the applications of topic modeling extend beyond the examples mentioned above. It aims to encompass diverse fields where efficient organization and analysis of textual data are essential.

Latent Dirichlet Allocation, commonly known as LDA[2], is a topic modeling algorithm that offers an automated solution. LDA aims to discover the underlying topics in the corpus and the corresponding proportions of all topics in each document using some key assumptions. Firstly, the key assumption is documents with similar topics have similar words. Secondly, words that belong to the same topic tend to co-occur in similar contexts. Thirdly, every document is viewed as a probability distribution over multiple topics. Finally, LDA aims to devise topic models by disregarding the sequence of words in a document, prioritizing the frequency of the words of interest.

A topic model can be inferred using either:

- Variational Inference[2]
- Gibbs Sampling[3]

Variational Inference is an example of approximating the true posterior probability distribution of the topics present in the corpus. In the context of LDA, variational inference helps in efficiently estimating the topic distributions for documents and the word distributions for topics, allowing for faster analysis of text corpora. This method transforms the difficult problem of integrating over the posterior distribution into an optimization problem, making it more computationally feasible to uncover the latent thematic structure in the text corpus. Whereas, Gibbs sampling is an algorithm that successively samples conditional distributions of the latent variables to the true distribution. Gibbs Sampling follows the method of Monte Carlo Markov Chain sampling, which samples from a probability distribution. While Gibbs sampling offers a more asymptotically exact implementation of the true posterior distribution[4], it can be computationally slower than variational inference and is better suited to smaller datasets.

We further extend the knowledge of inferring parameters for topic models by incorporating rough k-means clustering for preprocessing. Rough k-means clustering extends the traditional k-means algorithm by incorporating rough sets to handle uncertainty and vagueness in data clustering. This modification addresses the limitations of standard k-means, particularly with data that has unclear cluster boundaries. It introduces flexibility into the clustering process by allowing data points to have partial membership in multiple clusters. Rough k-means employ lower and upper approximations, where data points can belong to multiple clusters. This helps capture the ambiguity inherent in topic modeling algorithms like Latent Dirichlet Allocation, where a word or document could belong to multiple topics.

1.1 Motivation

The motivation for this dissertation stems from the desire to achieve faster convergence to the true posterior distribution using Gibbs sampling. While existing methods leverage k-means to enhance convergence[5] to the true posterior distribution, our approach introduces a method called **RKMS-LDA**. The aim is to further converge to the intended distribution more efficiently by

leveraging rough k-means because it aligns with the fundamental concept of LDA: words can belong to multiple topics with different probabilities. Rather than confining words to a single center, rough k-means allow for a more flexible assignment, reflecting the probabilistic nature of topic distributions.

1.2 Problem Statement

We aim to design an algorithm called RKMS-LDA that leverages Rough k-means Clustering to enhance the initialization phase of topic assignments in Latent Dirichlet Allocation (LDA). By avoiding randomization and utilizing soft clustering, RKMS-LDA seeks to make informed initial guesses about word-topic assignments. This approach is intended to facilitate faster convergence to the true posterior distributions, improving the modeling of both topic-word and document-topic distributions.

1.3 Dissertation Structure

- **Chapter 2: Literature Review**
  This chapter explores various topic models, with a particular focus on Latent Dirichlet Allocation (LDA), especially the Gibbs sampling technique, which is used for parameter estimation in this model. This review sets the foundation for understanding the advancements proposed in subsequent chapters.

- **Chapter 3: Rough k-means Clustering**
  This chapter introduces Rough k-Means clustering, explaining its principles and methodology. The chapter covers how Rough k-Means incorporates the concepts of lower and upper approximations to create soft clusters, which is essential for the preprocessing step in the new algorithm.

- **Chapter 4: Formulation of RKMS-LDA**
  This chapter combines the knowledge of collapsed Gibbs sampling and the fuzzy attributes of Rough k-Means clustering to develop a new algorithm: RKMS-LDA. The formulation process is demonstrated in a detailed manner.

- **Chapter 5: Experiments and Results**
  This chapter presents a comparative analysis of RKMS-LDA and traditional LDA using collapsed Gibbs sampling. The performance of both

algorithms is evaluated on five different datasets, providing a comprehensive assessment of the effectiveness of RKMS-LDA.

- **Chapter 6: Conclusion and Future Work**
  The final chapter discusses the conclusions drawn from the research and suggests ways to further improve the RKMS-LDA model. Furthermore, the implementation of distributed approaches using platforms like Apache Spark is proposed, as well as extending the model to handle streaming data, in addition to batch data.

Chapter 2

Literature Review

As our knowledge expands exponentially through blogs, books, images, news, scientific articles, social networks, video, web pages, and audio, it becomes crucial to develop topic models to organize, search, and comprehend this immense volume of information. Topic modeling is a probabilistic method designed to discover the underlying semantic structure of a collection of documents through a hierarchical Bayesian analysis of the text. Additionally, topic models offer a powerful method for representing documents as a probabilistic mixture of topics. Each document is expressed as a weighted combination of these latent topics, where the weights represent the document's thematic composition. A practical example is the analysis of customer reviews, where the use of topic models proves essential. Firstly, it can unveil latent thematic structures within the reviews, uncovering topics like product quality, customer service experience, or specific features. Secondly, by representing each review as a probability distribution of these topics, the classification of reviews based on their dominant themes can be done. Likewise, the applications of topic models extend to various other fields, such as analyzing scientific literature to identify emerging research trends, processing legal documents to categorize case law themes, identifying hidden structures in gene expression data, and many more.

2.1 Topic modeling Algorithms

Topic modeling algorithms have continuously evolved to provide more fine-tuned models that effectively uncover complex thematic structures within large corpora of documents.

2.1.1 Latent Semantic Analysis (LSA)

LSA models documents as a Bag-of-Words (BOW). This assumes each document is a simple collection of individual words, and the meaning is derived solely from word frequency. Additionally, LSA assumes documents are a mixture of all latent topics. Latent Semantic Analysis (LSA) was the first algorithm to do this [6].

LSA requires a document-term matrix (DTM) constructed from the text corpus. This matrix is essential as it captures the co-occurrence of words throughout the entire document collection. An example of a DTM is provided below.

Example:

Doc 1: "Smartphone has a sleek design."
Doc 2: "New smartphone features advanced camera."
Doc 3: "Sleek design and advanced features."

Document	smartphone	has	a	sleek	design	new	features	advanced	camera	and
Doc 1	1	1	1	1	1	0	0	0	0	0
Doc 2	1	0	0	0	0	1	1	1	1	0
Doc 3	0	0	0	1	1	0	1	1	0	1

Table 2.1: Sample Document-Term Matrix

Each cell in the DTM matrix contains a count of how often a word occurs in each document. However, using word-counts is not a very practical solution as it does not account for the significance of each word in the document. Instead of using word counts in the document-term matrix (DTM), a **tf-idf** score[7] is employed to measure the significance of each word in the document.

$$\text{TF\_IDF}(i,j) = \text{term\_frequency}(i,j) \times \log \frac{N}{\text{df}_j} \tag{2.1}$$

where:

- N is the number of documents.
- term_frequency(i,j) is the number of occurrences of term i in document j.
- df_j is the number of documents containing the term j.

LSA uses Singular Value Decomposition (SVD), a dimensionality reduction technique, to decompose the DTM into three separate matrices: document-topic, topic importance, and topic-term. While this approach offers a foundation for topic modeling, algorithms such as Latent Dirichlet Allocation (LDA) have emerged as a more powerful alternative. Unlike LSA, which assumes all documents contain all topics to some degree, LDA offers a probabilistic framework where documents are modeled as mixtures of a predefined number of latent topics, leading to more fine-tuned results.

2.1.2 Probabilistic Latent Semantic Analysis (pLSA)

Probabilistic Latent Semantic Analysis (pLSA) mitigates the shortcomings of LSA's Singular Value Decomposition (SVD) by introducing a probabilistic framework[2]. This probabilistic model is formulated using some key assumptions:

- Each document (d) is assumed to be a mixture of latent topics (z), represented by the probability P($z|d$).
- The probability of a specific word w being drawn from a topic z is denoted by P($w|z$).

$$P(D, W) = \sum_{z} P(D) \cdot P(z \mid D) \cdot P(w \mid z) \tag{2.2}$$

Here, $P(D, W)$ represents the joint probability of observing a document d containing a word w.

Probabilistic Latent Semantic Analysis (pLSA) uses Expectation-Maximisation to refine $P(z \mid D)$ and $P(w \mid z)$ probabilities to converge to more meaningful and thematic representations of the corpus[2]. Expectation-Maximization is an iterative optimization method designed to adjust model parameters, enabling them to approximate its document collection closely. In the E-step, hidden variable assignments are estimated, while the M-step updates model parameters to maximize the likelihood based on those estimates.

The next topic modeling algorithm, **Latent Dirichlet Allocation** (LDA) extends pLSA by introducing a Dirichlet distribution over the topics and providing a more interpretable and robust framework for discovering latent topics in the text corpus.

2.1.3 Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) serves as an extension to pLSA. Unlike pLSA, LDA is an example of a generative probabilistic model for collections of documents. The working of LDA is based on two key assumptions:

- Topics are probability distributions over words.
- Documents are probability distributions over topics.

An example can best illustrate this. A topic related to "Technology" would be a probability distribution of various words across the corpus. For example, words such as "computer", "internet", and "software" would have higher probabilities and would be more central to the topic. Subsequently, a document that has "AI applications in healthcare" need not pertain only to the "Technology" topic but also to the "Healthcare" topic. Multiple topics might be included in a single document with varying degrees of probability.

Before getting into the generative process of LDA, it is essential to understand a crucial element of the model: the **Dirichlet** prior.

The Dirichlet distribution is a multivariate generalization of the Beta distribution [8]. The Beta distribution is a continuous probability distribution that represents the probability of an event occurring between 0 and 1. It plays a pivotal role in various statistical applications, particularly in topic modeling. It offers a powerful way to model random probability mass functions (PMFs) for finite sets of outcomes. Essentially, it's a probability density function that describes the probabilities of probabilities themselves. Formally, a k-dimensional Dirichlet distribution, denoted by $\text{Dir}(\alpha)$, is parameterized by a vector $\alpha = [\alpha_1, \alpha_2, \ldots, \alpha_k]$, where each α_i is a positive real number. This distribution governs a k-dimensional random variable $\theta = [\theta_1, \theta_2, \ldots, \theta_k]$ that lies within a special geometric region called the $(k-1)$-simplex. In the context of the Dirichlet distribution, the k-dimensional simplex ensures that all elements in the random variable θ are non-negative ($\theta_i > 0$) and sum to 1.

The probability density function of a k-dimensional Dirichlet distribution with parameter α is given by:

$$p(\theta \mid \alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{k} \theta_i^{\alpha_i - 1} \tag{2.3}$$

Here, Γ represents the gamma function. This equation defines the probability of a specific k-dimensional probability distribution (θ) given the Dirichlet prior parameter α.

The influence of the Dirichlet Distribution is such that it embodies a crucial assumption about how documents are generated in the context of topic modeling. By influencing the topic proportions within each document, the Dirichlet prior shapes our perception of the underlying thematic structure. This can be best illustrated by an example that shows how the Dirichlet prior shapes our understanding of documents in the context of topic modeling.

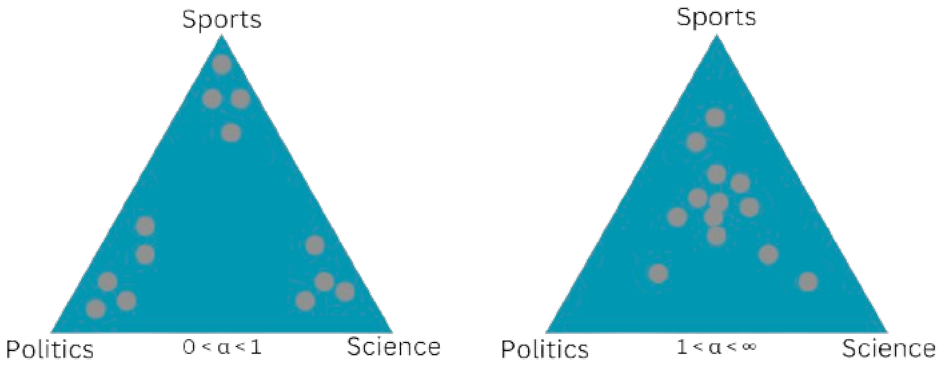


Figure 2.1: Example of Dirichlet distributions for a 3-topic corpus.

For example, assuming a scenario where we have three potential topics: science, politics, and sports. When α is set to a value less than 1 and greater than 0, it encourages sparsity. Documents tend to cluster towards the corners of a triangular space, reflecting a strong affinity for a single dominant topic. Conversely, an α greater than 1 promotes a more even spread of topics. Documents gravitate towards the center of the space, exhibiting a more balanced mixture of topics.

Having understood the role of the Dirichlet prior, the three-level hierarchical generative process of Latent Dirichlet Allocation (LDA) is as follows:

1. **Document-Topic Proportions:**
   A document-specific topic distribution (θ_d) is drawn from a Dirichlet distribution with parameter α.

$$\theta_d \sim \text{Dir}(\alpha)$$

2. **Picking a topic from the Document-Topic Distribution:**
   From each document, A topic (z_w) is chosen for the word based on the document's topic distribution (θ_d) using a multinomial distribution:

$$z_w \sim \text{Multinomial}(\theta_d)$$

3. **Picking a word:**
   A word (w) is generated based on the chosen topic (z_w) and another Dirichlet prior parameter β, which represents the topic-word distribution:

$$w \sim \text{Multinomial}(\text{p}(w|z_w, \beta))$$

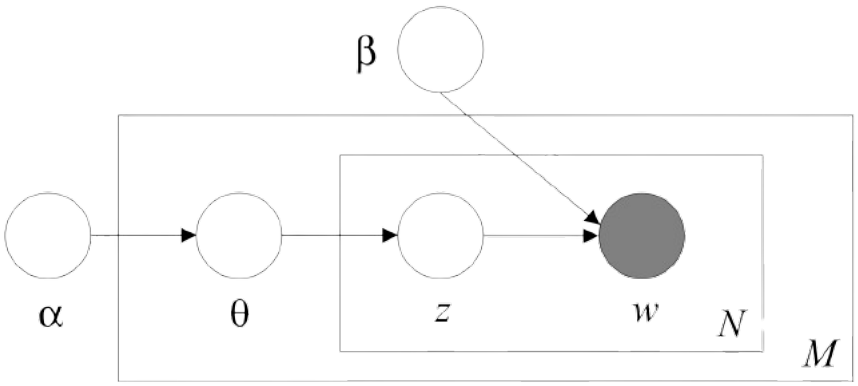


Figure 2.2: The generative process of Latent Dirichlet Allocation.

Based on the generative model, the constructed document that most accurately replicates the original document using the appropriate parameters helps infer the model. However, the document-topic and topic-word distributions are mathematically intractable to infer using this model.

$$p(\theta, z \mid w, \alpha, \beta) = \frac{p(\theta, z, w \mid \alpha, \beta)}{p(w \mid \alpha, \beta)} \tag{2.4}$$

To curb this intractability, Variational Inference is used to approximate the model for the text corpus.[2]

1. **Variational Inference**
   Variational Inference offers a solution by approximating the θ and z parameters. The graphical model (see Figure 2.2) has a problematic coupling between θ and β. This complex interplay between variables makes it mathematically intractable to calculate the exact posterior distribution. The graphical model therefore is changed and the edges between θ and β are dropped.

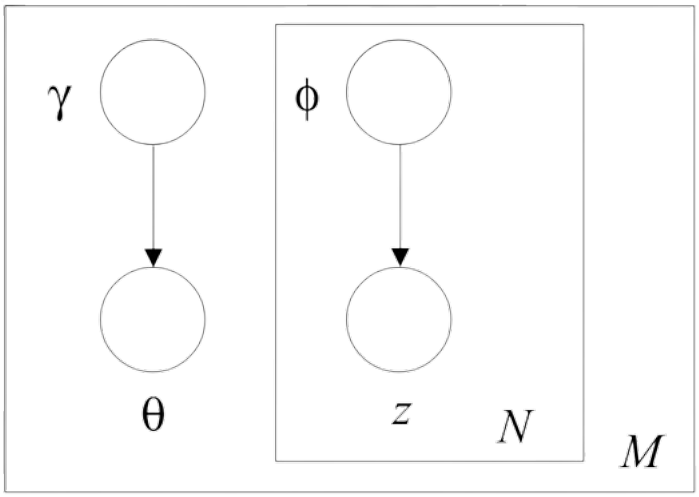


Figure 2.3: Graphical model to approximate the posterior distributions

Variational Inference introduces a variational distribution $Q(\theta, z)$ that approximates the true posterior distribution $p(\theta, z \mid D)$. This variational distribution is a simpler and more tractable form. The approximation is done by lessening the difference between $p(\theta, z \mid D)$ and $Q(\theta, z)$ using the Kullback-Leibler (KL) divergence method between both distributions. This allows for an attainable approximation of the true posterior distribution. An extension to using KL-divergence was to incorporate Expectation-Maximisation to infer the posterior. Variational EM, iteratively leverages the EM framework but uses the variational distribution instead of the true posterior in the E-step.

While Variational Inference is expected to be faster, other inference algorithms, such as Gibbs sampling can potentially converge to a distribution closer to the true posterior.

2. **Gibbs Sampling**
Gibbs Sampling is a Markov Chain Monte Carlo (MCMC) method to approximate the posterior distribution. MCMC methods are straightforward and intuitive approaches for inferring values from an unknown distribution and utilizing those values for further analysis [9]. At its core, Gibbs sampling iteratively updates the hidden variable assignments by considering the conditional probabilities of each variable given the current state of all other variables. An example can illustrate this.

Doc 1	**Doc 2**	**Doc 3**
Athlete	Physics	Experiment
Runs	Calculates	Calculates
Quantum	Jump	Athlete
Mechanics	Athlete	Speed
Speed	Air	Limit

With the current topic assignments for all other words (except "Athlete"), we calculate the conditional probability of "Athlete" belonging to each topic (Sports & Science).

- We assume the color red is for "Sports" and the color blue is for "Science"
- Consider the current topic assignments of "Runs," "Quantum, "Speed", and "Mechanics" in Document 1.
- We look at the topic assignments in Documents 2 and 3 for words related to sports (e.g., "Physics" and "Jump") and science (e.g., "Athlete," "Limit," and "Calculates").
- Based on these assignments and word-topic distributions, calculate the conditional probability of "Athlete" being a Sports term or a Science term.

As this process is repeated for every word in the corpus over several iterations, the topic assignments at the end are likely to resemble the following:

Doc 1	Doc 2	Doc 3
Athlete	Physics	Experiment
Runs	Calculates	Calculates
Quantum	Jump	Athlete
Mechanics	Athlete	Speed
Speed	Air	Limit

After multiple iterations, words pertaining to specific topics are:

- **Sports:** {Athlete, Runs, Jump, Speed}
- **Science:** {Quantum, Mechanics, Calculates, Physics, Air, Experiment, Limit, Speed}

Subsequently, the document-topic distributions are: {Sports, Science}

- **Doc 1:** {0.6, 0.4}
- **Doc 2:** {0.4, 0.6}
- **Doc 3:** {0.2, 0.8}

In LDA notation, this translates to sampling the topic assignment (z_i) for word i, given the topics assigned to all other words (z_j for $j \neq i$), the document the word belongs to, and the current estimates of the topic-word distribution (β). However, standard Gibbs Sampling can become computationally expensive, especially for large datasets.

$$P\left(z_{idn} = 1 \mid w, \beta^{(t+1)}\right) = \frac{\theta_{di}\beta_{iw_{dn}}^{(t+1)}}{\sum_{d=1}^{M} \theta_{di}\beta_{iw_{dn}}^{(t+1)}} \tag{2.5}$$

This equation calculates the probability of a specific word w belonging to a particular topic z_i within document d at iteration $t+1$ of the Gibbs sampling process. θ_{di} represents the weightage or mixture of various topics that make up document d. $\beta_{iw}^{(t+1)}$ captures the weightage of how a particular word w tends to appear within a specific topic i. The summation in the denominator is worked out by iterating over all documents M in the corpus. It calculates the probability of word w belonging to any possible topic i in document d at iteration $t+1$.

This equation represents a single update step within the Gibbs sampling algorithm for LDA. The algorithm iterates through all words in the corpus, performing this calculation for each word and each possible topic assignment. The algorithm needs to iterate through all words in the corpus for each update step. Each variable is sampled from its conditional distribution while keeping the other variables fixed. This is where **collapsed** Gibbs Sampling comes in, by integrating the distributions out of the equation. By collapsing these variables, the sampling process becomes more efficient by reducing the dimensionality of the sampling space.

$$P(z_i = j \mid z_{-i}, w) \propto \frac{n_{w_i} + \beta}{n_{-i,j}^{(\cdot)} + V\beta} \frac{n_{-i,j}^{(d_i)} + \alpha}{n_{-i}^{(d_i)} + K\alpha} \tag{2.6}$$

n_{w_i} is the number of times word w_i was related to topic j excluding the current instance, denoted as $-i, j$. $n_{-i,j}^{(\cdot)}$ represents the number of times all other words were related to topic j. $n_{-i,j}^{(\cdot)} + V\beta$ is the normalization factor for the word-topic assignments. $(n_i^d)_{-i,j}$ is the number of times topic j was related to document d_i excluding the current instance, denoted as $-i, j$. $(n_i^d)_{-i}$ is the number of times all other topics were related to document d_i. V is the number of words in the vocabulary and K is the number of topics. [10]

After a fixed number of iterations, we arrive at counts within the topic-word and document-topic matrices. Using these counts, we can directly infer the posterior distributions in the provided equations.

$$\hat{\phi}_j^{(w)} = \frac{n_j^{(w)} + \beta}{n_j^{(\cdot)} + V\beta} \tag{2.7}$$

$$\hat{\theta}_j^{(d)} = \frac{n_j^{(d)} + \alpha}{n_{(\cdot)}^{(d)} + K\alpha} \tag{2.8}$$

$n_j^{(d)}$ reflects how many words in document d belong to topic j, while $n_{(\cdot)}^{(d)}$ simply counts the total words in document d. Similarly, $n_j^{(w)}$ captures the total number of times word w is assigned to topic j across all documents, and $n_j^{(\cdot)}$ represents the overall word count assigned to topic

j in the entire corpus.

Equation 2.7 is the Bayesian estimation of the distribution of words in a topic. Equation 2.8 is the Bayesian estimation of the distribution of topics in a document[10].

Collapsed Gibbs sampling allows us to focus solely on inferring word-topic and document-topic distributions directly from topic assignments[11] without explicitly mentioning them in the equations.

2.1.4 Future Models

Further advancements were made by constructing topic models using Latent Dirichlet Allocation (LDA). Despite its effectiveness, the bag-of-words model, the necessity of specifying the number of topics (k) beforehand, and other limitations indicate room for improvement in achieving more accurate and meaningful results. While LDA remains a valuable tool, other approaches such as using **BERTopic** leverage pre-trained language models (like BERT)[12] to identify semantically coherent topics. Additionally, complex semantics are also being captured using word embeddings like **Word2Vec**[12] to reveal semantic relationships between words as well as based on co-occurrences in sentences. However, the core of this dissertation focuses on Latent Dirichlet Allocation (LDA) and the attempt to infer the model faster using a slight modification of Collapsed Gibbs Sampling.

2.2 Coherence Measures

Topic models like Latent Dirichlet Allocation (LDA) provide valuable insights by uncovering the thematic structure within document collections. The model outputs two key distributions: topic-word distributions and document-topic distributions. However, how well do these distributions represent the true thematic structure of the corpus? This is where a **coherence measure** comes into play.

2.2.1 Introduction

Imagine two pieces of information that reinforce each other, like "cricket" and "bat." We can say they are coherent because they frequently appear in similar contexts and mutually support one another. This concept of coherence is crucial in topic modeling, a technique that uncovers hidden thematic structures within document collections. Coherence measures address the challenges associated with human subjectivity and the inherent expense of capturing human interpretations[13]. The goal is to develop automated metrics that closely align with the golden standard of human judgment and assess the quality of the topics discovered by the model[14]. Researchers have explored various methods for constructing coherence measures and employed a **four-stage process** to achieve this objective[13].

2.2.2 Framework for Coherence Measures

The coherence of a set of words measures how well individual words or subsets of them fit and relate to each other. The framework utilizes a four-step process. First, it breaks down the set of words associated with a topic into smaller subsets. This allows for a more granular analysis of word co-occurrence within the topic. Next, a probability measure is estimated for each subset. This step involves determining how likely it is to encounter these words together, considering various approaches like examining their co-occurrence within the entire document, specific sentences, or using a sliding window technique across the document. Following this, a confirmation measure is applied. This measure assesses how well the words within each subset support each other thematically, based on the calculated probabilities. Finally, the framework aggregates the results from the confirmation measure across all subsets, producing a single coherence score for the entire topic. This score reflects the overall thematic coherence of the topic, with higher scores indicating a stronger and more interpretable topic. We will delve into each of these stages in detail point by point[13].

(a) **Segmentation**:
In topic modeling, segmentation plays a critical role in calculating topic coherence. This is the first and foremost phase of the topic coherence framework. This process involves breaking down the

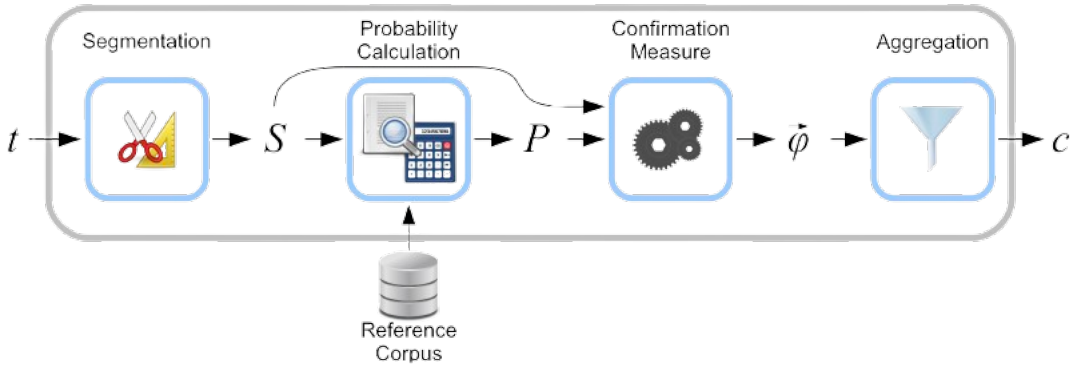


Figure 2.4: The four-step framework of formulating a coherence measure.

top words (tokens) extracted from each topic into smaller units. All possible subsets (word sets) of these tokens form the entire sample space of subsets used for coherence calculations. The specific segmentation technique employed determines the nature of these subsets. Different confirmation measures utilize these subset pairs (two subsets within the word set) uniquely. The core idea behind pairing subsets is to assess how well the first subset supports the second thematically. For example, consider the topic words: "Data," "Analysis," "Statistics," and "Trends." The segmentation process aims to quantify the support between these words. Intuitively, "Analysis" likely provides stronger support for "Data" compared to "Trends."

$W^{'}$ and W^{*} are word sets evaluated in segmentation techniques to determine how well W^{*} supports the existence of $W^{'}$. Based on the choice of coherence measure, we choose an appropriate segmentation technique.

$$S_{\text{one, one}} = \left\{(W^{'}, W^{*}) \mid W^{'} = \{w_i\};\ W^{*} = \{w_j\};\ w_i, w_j \in W;\ i \neq j\right\}$$

For instance, based on the example earlier, in a one-one segmentation word pairs would look like this:

($W^{'}$ ={'Data'}, W^{*} ={'Statistics'})
($W^{'}$ ={'Analysis'}, W^{*} ={'Trends'})

$$S_{\text{one, all}} = \left\{(W^{'}, W^{*}) \mid W^{'} = \{w_i\}; w_i \in W;\ W^{*} = W \setminus \{w_i\}\right\}$$

Similarly, in a one-all segmentation word pairs would look like this:

($W^{'}$ ={'Data'}, W^{*} ={'Statistics', 'Trends', 'Analysis})
($W^{'}$ ={'Analysis'}, W^{*} ={'Trends', 'Data', Statistics'})

$$S_{\text{one, any}} = \left\{(W^{'}, W^{*}) \mid W^{'} = \{w_i\}; w_i \in W;\, W^{*} \subseteq W \setminus \{w_i\}\right\}$$

The word pairs in a one-any segmentation would look like this:
($W^{'}$ ={'Data'}, W^{*} ={'Statistics', 'Trends', 'Analysis})
($W^{'}$ ={'Analysis'}, W^{*} ={'Trends', 'Data', Statistics'})
($W^{'}$ ={'Data'}, W^{*} ={'Statistics', 'Trends'})
($W^{'}$ ={'Trends'}, W^{*} ={'Data'})

$$S_{\text{any, any}} = \left\{(W^{'}, W^{*}) \mid W^{'}, W^{*} \subseteq W;\, W^{'} \cap W^{*} = \emptyset\right\}$$

The word pairs in an any-any segmentation would look like this:
($W^{'}$ ={'Data'}, W^{*} ={'Statistics', 'Trends', 'Analysis})
($W^{'}$ ={'Statistics', 'Trends'}, W^{*} ={'Data'})
($W^{'}$ ={'Trends'}, W^{*} ={'Data'})
($W^{'}$ ={'Trends', 'Analysis'}, W^{*} ={'Data', 'Statistics'})

(b) **Probability Estimation**:
The probability estimation phase plays a crucial role in calculating topic coherence. Here, we aim to estimate how often specific word sets co-occur within a fixed window size in the underlying text corpus. The chosen window size dictates the approach used for probability estimation. There are various ways of estimating these probabilities:

- Boolean Document ($\mathcal{P}_{bd}$): This approach calculates the probability by dividing the number of documents containing the word of interest by the total number of documents in the corpus.
- Boolean Paragraph ($\mathcal{P}_{bp}$): This approach calculates the probability by dividing the number of paragraphs containing the word of interest by the total number of paragraphs in the corpus.
- Boolean Sentence ($\mathcal{P}_{bs}$): This approach calculates the probability by dividing the number of sentences containing the word by the total number of sentences in the corpus.

- Boolean Sliding Window ($\mathcal{P}_{sw}$): This method employs a sliding window of a predefined number of tokens that scans the entire corpus. Each window is a virtual document. It is a Boolean document on a fixed number of tokens for a document.

(c) **Confirmation Measure**:
A confirmation measure takes a single pair $S_i = (W^{'}, W^{*})$ of words and computes how well W^{*} supports $W^{'}$. There are two types of measures:

- **Direct Confirmation Measures**:
Direct coherence measures simply compare two-word sets head-to-head. There are several formulae developed over time with each formula deeming a different approach to calculate how well a conditioning word set W^{*} supports $W^{'}$.

  i. **Difference Measure (m_d)**:

$$m_d(S_i) = P(W_0 \mid W^*) - P(W_0)$$

  ii. **Ratio Measure (m_r)**:

$$m_r(S_i) = \frac{P(W_0, W^*)}{P(W_0) \times P(W^*)}$$

  iii. **Likelihood Measure (ml)**:

$$m_l(S_i) = \frac{P(W_0 \mid W^*)}{P(W_0 \mid \neg W^*) + \epsilon}$$

  iv. **Log Likelihood Measure** (m_{ll}):

$$m_{ll}(S_i) = \log\left(\frac{P(W_0 \mid W^*) + \epsilon}{P(W_0 \mid \neg W^*) + \epsilon}\right)$$

  v. **Log Ratio Measure** (m_{lr}):

$$m_{lr}(S_i) = \log\left(\frac{P(W_0, W^*) + \epsilon}{P(W_0) \times P(W^*)}\right)$$

  vi. **Normalized Log Ratio Measure** (m_{nlr}):

$$m_{nlr}(S_i) = m_{lr}(S_i) - \log(P(W_0, W^*) + \epsilon)$$

vii. **Log Conditional Measure (m_{lc})**:

$$m_{lc}(S_i) = \log\left(\frac{P(W_0, W^*) + \epsilon}{P(W^*)}\right)$$

- **Indirect Confirmation Measures**:
  Unlike direct coherence measures that operate in a one-to-one comparison, focusing solely on how well word set $W^{'}$ supports word W^*, indirect measures address complex nuances in language. They recognize that some related words, like "Cadbury" and "Hershey's" which are brands of chocolate, might not directly co-occur but still share a strong conceptual link. Indirect confirmation measures go beyond surface-level co-occurrence, aiming to capture these deeper connections by analyzing how words from different sets relate to a broader concept, even if they don't appear in the same immediate context.
  Given context vectors $\vec{u} = \vec{v}(W_0)$ and $\vec{w} = \vec{v}(W^*)$ for the word sets of a pair $S_i = (W_0, W^*)$, indirect confirmation is computed as vector similarity. [13]

  $$\text{indirect\_confirmation}(S_i) = \text{similarity}(\vec{u}, \vec{w})$$

  Basically, an indirect confirmation for a word set is built based on the sum of all direct confirmations for all possible pairs of words in the word set. There are several types of similarity measures:

  $$s_{\cos}(\vec{u}, \vec{w}) = \frac{\sum_{i=1}^{|W|} u_i \cdot w_i}{\|\vec{u}\|_2 \cdot \|\vec{w}\|_2}$$

  $$s_{\text{dice}}(\vec{u}, \vec{w}) = \frac{\sum_{i=1}^{|W|} \min(u_i, w_i)}{\sum_{i=1}^{|W|} (u_i + w_i)}$$

  $$s_{\text{jac}}(\vec{u}, \vec{w}) = \frac{\sum_{i=1}^{|W|} \min(u_i, w_i)}{\sum_{i=1}^{|W|} \max(u_i, w_i)}$$

(d) **Aggregation**:
After calculating individual coherence scores, the next step is to combine them into a single, unified metric. We have options like mean, median, geometric mean, and harmonic mean. The choice depends on the specific context and the characteristics of our coherence framework. Tailoring the aggregation method ensures it

best reflects the underlying relationships within the data and delivers the most meaningful score for our analysis.

These four steps put together give us our coherence framework. Some very well-known coherence measures are as follows:

- $C_V = S^{\text{one}}_{\text{one}} \times \mathcal{P}_{\text{sw(110)}} \times \tilde{m}_{\text{cos(nlr,1)}} \times \sigma_a$
- $C_{UMass} = S^{\text{pre}}_{\text{one}} \times \mathcal{P}_{\text{bd}} \times m_{lc} \times \sigma_a$
- $C_{UCI} = S^{\text{one}}_{\text{one}} \times \mathcal{P}_{\text{sw(10)}} \times m_{lr} \times \sigma_a$
- $C_{NPMI} = S^{\text{one}}_{\text{one}} \times \mathcal{P}_{\text{sw(10)}} \times m_{nlr} \times \sigma_a$

In Chapter 5, we made use of the C_v coherence measure to validate the quality of our topic model. The next chapter explores the potential of rough k-means clustering. We investigate how by combining it with Gibbs sampling, we can develop an efficient and robust model for inferring topics.

Chapter 3

Rough k-Means Clustering

As discussed earlier, topic models identify latent themes in a collection of documents. For example, a topic related to "health" might have words like "doctor", "diet", and "disease" included. While these words are used in very similar contexts, a word like "diet" could be part of a document discussing healthy life choices, whereas "doctor" and "disease" could be part of a document discussing diseases. This was already discussed in Latent Dirichlet Allocation, and will further extend to leveraging soft clustering to preprocess text data. This is where Rough K-means clustering comes into play. This technique leverages the power of rough set theory to create more nuanced clusters, allowing documents to fall within a spectrum of membership based on the relevance of the topic. In the following sections, we'll delve deeper into rough k-means clustering, exploring its core principles and the foundation provided by rough set theory.

3.1 Introduction to Rough Set Theory

The foundation of rough set theory lies in the recognition that data points within an information system may not possess complete or definitive information. Rough sets address this by acknowledging varying degrees of information associated with each data point[15].

For example, some days are undeniably sunny, others might be partly cloudy with some sunshine. Traditional sets would struggle with such ambiguity. The rough set theory addresses this by creating a lower approximation containing data points definitively classified as sunny. An upper approximation encompasses all data points that could potentially be sunny, including those with some degree of uncertainty (e.g., partly cloudy days). The space between these two approximations represents the boundary region, acknowl-

edging the existence of ambiguity that classical set theory might ignore.

Imagine a Venn diagram with two overlapping circles representing Sets A and B. The data points entirely within Set A depict the lower approximation. The data points subscribing to both sets A and B, including the overlap region, represent the upper approximation.

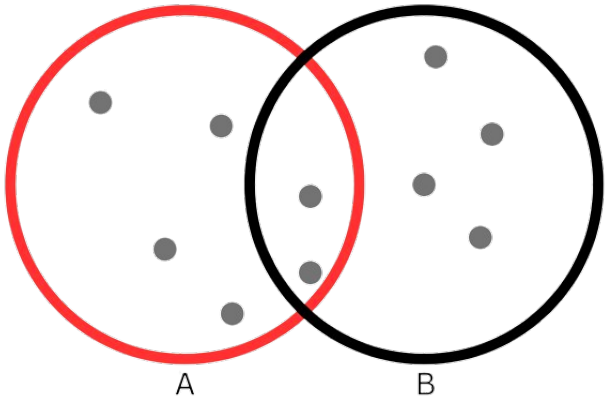


Figure 3.1: A Venn Diagram resembling the lower and upper approximations in Rough Sets.

Rough set theory provides a powerful mathematical framework for handling vague and ambiguous information within the universe of discourse[16]. This approach is particularly valuable when dealing with incomplete or uncertain data, where not all information is readily available. Unlike traditional set theory, which requires data points to definitively belong to a set or not, rough set theory introduces the concepts of lower and upper approximations.

Traditional clustering techniques often assume data points belong definitively to a single cluster. However, the complexity of the real world is such that imprecise, uncertain, or incomplete information are available in the data. This can lead to data points existing in boundary regions between clusters, making strict assignment in crisp boundaries difficult. Rough k-means clustering addresses this by leveraging concepts from rough set theory. The concepts of lower and upper approximations handle data points by assigning them to multiple clusters based on membership degrees. Clustering algorithms, such Fuzzy C-Means Algorithm, Min-max Rough clustering, make use of rough set concepts like indiscernibility relations to group similar data points and knowledge granules to represent clusters with varying levels of

granularity, leading to robust clustering results even in the presence of uncertainty.

This philosophy of embracing "grey areas" allows for a more intricate analysis. Rough sets enable the characterization sets of objects based on attribute values in an information table. This allows for discovering relationships between attributes, and identifying if one attribute always or sometimes predicts another. Additionally, rough sets can identify significant attributes that contribute most to the characterization of a set.

A key advantage is that rough set theory requires minimal prior information compared to other approaches like statistics which rely on probabilities[15]. This makes it a valuable tool for exploratory data analysis where the data characteristics might not be fully understood beforehand.

For instance, rough set theory finds applications in various domains, including medical diagnosis where patient classification relies on potentially incomplete medical history, fraud detection in the banking sector where ambiguous transaction patterns need identification, and recommender systems that provide suggestions based on imprecise or uncertain data.

3.2 Rough k-means clustering

In machine learning, clustering is a machine learning technique to group data together based on their similarities most frequently in larger datasets. By grouping similar data points together, clustering helps us uncover hidden patterns and relationships within datasets. Unlike supervised learning, the expense of labeling is eliminated and the lack of need for a predefined label allows for larger exploratory analysis.

A fundamental principle in evaluating clustering models is the notion of intra-cluster homogeneity and inter-cluster heterogeneity[17]. Ideally, data points within a cluster should be as similar as possible (homogeneous), while points belonging to different clusters should exhibit greater dissimilarity (heterogeneous). Clustering techniques can be broadly categorized as hierarchical or partitional (non-hierarchical). Partitional clustering, like k-means, creates distinct clusters without allowing data points to belong to multiple clusters simultaneously.

However, it's important to note that rough k-means clustering shares

some limitations with its parent algorithm, k-means. It relies on points as cluster representatives and the squared Euclidean distance as the dissimilarity measure, possibly leading to numerical instability due to high-dimenstional data.

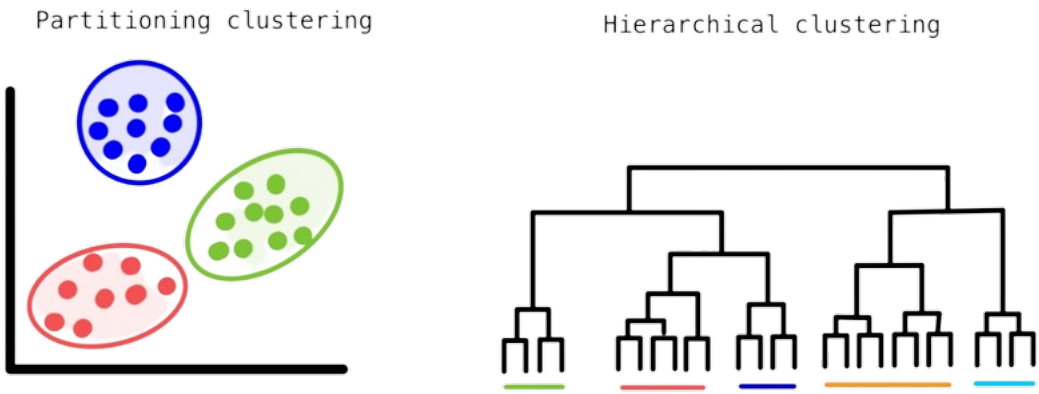


Figure 3.2: Partitional and Hierarchical clustering.

K-means clustering, a popular unsupervised learning algorithm, partitions data points into a pre-defined number of clusters. Here's a breakdown of its key steps:

- **Step-1**: K-means starts by randomly selecting K points from the dataset as centroids.
- **Step-2**: Each data point is assigned to the closest centroid. This could be the Euclidean or Manhattan Distance. This initial assignment forms the initial clusters.
- **Step-3**: The centroids are recalculated by taking the mean of the data points assigned to each cluster.
- **Step-4**: Repeat Steps 2 and 3 until a convergence point is reached. The converge point more often than not is to calculate the movement of the clusters from the previous iteration and to ensure it is as low as possible.

The methodology of Rough k-means is similar to k-means where initializing with k centroids randomly chosen from the data. However, the core difference lies in how data objects are assigned to clusters. Unlike the crisp

assignments of k-means, rough k-means allow for a more nuanced approach using "lower approximations" and "upper approximations". Objects can belong to multiple clusters simultaneously. If an object belongs to a cluster's lower approximation, it signifies a strong membership. The semantics are such that if an object belongs to the lower approximation, it belongs to the upper approximation of the same cluster and is not in the upper approximation of any other clusters. Conversely, an object in the upper approximation of at least two clusters indicates a weaker, borderline membership in those clusters. These objects are considered to be on the boundary between multiple clusters. The assignment of belonging to the lower and upper approximation of the object to the clusters is done using the following procedure:

- The distances between the given object x and all centroids are computed. Let $\vec{m}_h$ represent the nearest cluster centroid. Hence, the least distance is given by:

$$d_{n,h}^{\min} = d(\vec{X}_n, \vec{m}_h) = \min_{k=1,\ldots,K} d(\vec{X}_n, \vec{m}_k). \tag{3.1}$$

- Then x is assigned to the upper approximation of $\overline{C}_h$, the nearest cluster. Then the feasibility of x belonging to the upper approximations of other clusters C_k (where $k \neq h$) is arrived at using $relative\_dist\_threshold$. If $d(x, m_k)$ divided by $d(x, m_h)$ is less than the relative distance threshold, then x is assigned to $\underline{C}_k$.

- After this feasibility analysis, if x belongs to the upper approximation of more than one cluster, then x is not assigned to the lower approximation of any cluster. Otherwise, x is included in $\underline{C}_h$ (in the lower approximation of the closest cluster).

- In every iteration, every object's membership into lower or upper approximation is done using this procedure for each cluster C_k. $\underline{C}_k$ represents the lower approximation and $\overline{C}_k$ represents the upper approximation.

- Then the centroid of the clusters is updated using the weighted average of mean vectors of lower and upper approximations. For the k^{th} cluster, the centroid is updated as follows:

$$\vec{m}_k = w_l \sum_{\vec{X}_n \in \underline{C}_k} \frac{\vec{X}_n}{|\underline{C}_k|} + w_u \sum_{\vec{X}_n \in \overline{C}_k} \frac{\vec{X}_n}{|\overline{C}_k|} \tag{3.2}$$

  with $w_l + w_u = 1$

- Once the centroids are updated, the convergence criteria are met when there is no change in the cluster centroids in the next iteration and minimal movement in the assignment of data objects.
- In case of convergence, we obtain the cluster lower and upper approximations. We return l, C_l, $\underline{C}_l$, $\bar{C}_l$, and m_l(where $l = 1$ to k)[18].

Algorithm 1 `RoughKMeansClustering`

Input:

- Data matrix M
- Number of clusters k
- Maximum iterations max_iterations
- Relative distance threshold relative_dist_threshold

Output:

- Centroid vectors $\{c_1, c_2, \ldots, c_k\}$

Method:

Initialization:

Ensure each data object belongs to the lower approximation of multiple clusters, which also means it belongs to the upper approximation of the same cluster.

while not converged and iterations < max_iterations **do**
    for each data object d_i in M **do**
        Calculate distances to all centroids.
        Assign d_i to the lower approximation of the closest cluster.
        if the distance to the closest centroid is greater than relative_dist_threshold **then**
            Assign d_i to the upper approximation of that cluster.
        end if
    end for
    Ensure each cluster's lower approximation has at least one member.
    Update centroids:
    for each cluster **do**
        Calculate the mean vector of all data objects assigned to the cluster.
        Update the centroid to this mean vector.
    end for
    Check convergence criteria.
end while
return Centroid vectors $\{c_1, c_2, \ldots, c_k\}$

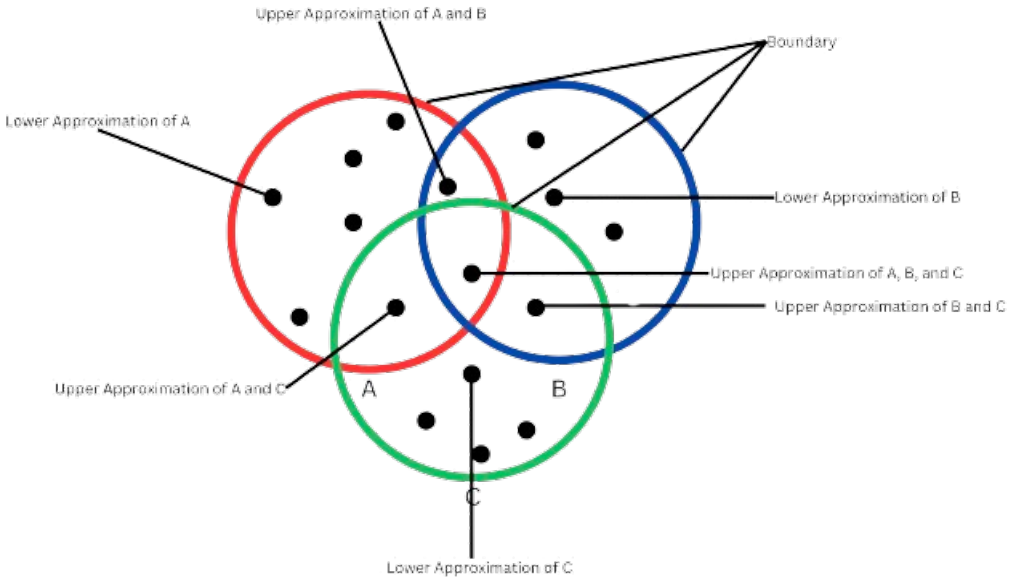


Figure 3.3: An example of Rough k-means clustering with k=3.

As mentioned earlier, traditional k-means assign data points definitively to single clusters with crisp boundaries, often failing to capture the inherent ambiguity of real-world data. Probabilistic measures, address this by possibly representing cluster centroids as probability distributions[5]. Similar to rough k-means' ability to handle ambiguity in cluster membership, Latent Dirichlet Allocation (LDA) uses the shared membership model in document-topic and word-topic assignments. This shared focus on handling vagueness makes rough k-means a valuable tool for preprocessing text corpora before applying LDA. This pre-processing step has the potential to mitigate the time required for LDA to converge and reach the true distribution of document-topic and word-topic relationships. The next chapter will delve deeper into this implementation, exploring how rough k-means can be integrated with the collapsed Gibbs sampling.

Chapter 4

Formulation of RKMS-LDA Algorithm

4.1 Motivation

Latent Dirichlet Allocation (LDA) operates using two primary probability distributions: topics over documents and words over topics. More often than not, document collections exhibit ambiguity, where documents or topics can be related to various themes or words to varying degrees. To address this, RKMS-LDA introduces a preprocessing step using rough k-means. Rough k-means, with its ability to handle partial memberships, allows us to refine the initial topic structure. We can cluster words based on their thematic similarity, resulting in k-clusters (k-topics) with varying degrees of membership for each word. This preprocessed information, incorporating the concept of partial membership, is then used for sampling in the subsequent iterations. Furthermore, standard LDA utilizes collapsed Gibbs sampling, where initial word-topic assignments are random before the sampling process begins. Instead of random initialization, RKMS-LDA leverages the partial membership information obtained through rough k-means to create a more informed initial assignment of words to topics. This can bring us closer to the true posterior distributions, ultimately leading to a more accurate and efficient topic modeling process.

4.2 Collapsed Gibbs Sampling

Preprocessing: The text corpus undergoes preprocessing to improve data quality. This includes removing complex regular expressions, punctuation, and stopwords. Additionally, corpus-specific stopwords might be tai-

lored based on the inference task. Empty rows are dropped, and only the text data column is retained for further processing. Finally, each document (row) is lemmatized and tokenized, resulting in a clean and ready-for-processing dataset.

There are two key phases to performing Gibbs Sampling to infer the LDA model.

- Initialise topic assignments for each word.
- Sample for n-iterations and update topic assignments gradually.

Let us focus on initializing topic assignments first. This process relies on several key elements defined by pre-specified Dirichlet priors (α and β).

- **Document-Topic Matrix** (n_{dk}): This matrix represents the number of words assigned to each topic (k) in each document (d).
- **Topic-Word Matrix** (n_{kw}): This matrix captures the number of times each word (w) is assigned to each topic (k).
- **Words per Topic** (n_k): This represents the total number of words assigned to each topic (k).
- **Individual Topic Assignments**: This variable keeps track of the current topic assignment for each word in the corpus

Algorithm 2 `initGibbs`

Input:

- α, β: Dirichlet priors
- n_{topics}: Number of topics
- vocab: Vocabulary
- docs: Documents

Output:

- n_{dk}: Document-topic counts
- n_k: Per-topic word counts
- n_{kw}: Topic-word counts
- docs_assignments: List of word assignments to topics for each document

Method:

Set α and β Dirichlet priors
$k \leftarrow n_{\text{topics}}$ ▷ Set number of topics
$V \leftarrow$ length(vocab) ▷ Set size of vocabulary
$M \leftarrow$ length(docs) ▷ Set number of documents
$n_{kw} \leftarrow \mathbf{0}_{(k,V)}$ ▷ Initialize topic-word matrix with zeros
$n_{dk} \leftarrow \mathbf{0}_{(M,k)}$ ▷ Initialize document-topic matrix with zeros
$n_k \leftarrow \mathbf{0}_k$ ▷ Initialize words per topic with zeros
Initialize empty list docs_assignments
for doc **in** docs **do**
    Initialize empty list assignments
    for word **in** doc **do**
        Generate random integer topic_index between 0 and k-1
        Assign word the topic corresponding to topic_index
        Append topic_index to assignments
    end for
    Append assignments to docs_assignments
end for
Update n_{kw} ▷ Initial topic-word counts
Update n_{dk} ▷ Initial document-topic counts
Update n_k ▷ Initial per-topic word counts
return n_{dk}, n_k, n_{kw}, docs_assignments

- The algorithm iterates through each document and for each word in the document, it assigns a random topic index between 0 and k-1.
- This random assignment process ensures that each word in each document is assigned a topic randomly from the available topic indices.

The core idea behind the sampling is that the loop iterates through each word and its corresponding topic assignment in the corpus holding all other topic assignments fixed. During each iteration, the algorithm samples a new topic assignment for the current word based on the conditional probability distribution considering the current topic assignments of all other words in the document and the overall topic distribution across the corpus. (Shown in Algorithm 4)

Algorithm 3 `conditionalProbability(position, d, k)`

Input:
  position: Column index of the word in the n_{kw} matrix
  d: Index of the document
  k: Number of topics
Output:
  prob: Array of conditional probabilities for each topic
Method:
Initialize array prob of size k
for $i = 0$ to $k - 1$ **do**
    Calculate term_1 as:
      numerator_1 $\leftarrow n\_kw[i, \text{position}] + \beta$
      denominator_1 $\leftarrow \sum$ all entries in row i of $n\_kw + V \cdot \beta$
      term_1 $\leftarrow \frac{\text{numerator\_1}}{\text{denominator\_1}}$
    Calculate term_2 as:
      numerator_2 $\leftarrow n\_dk[d, i] + \alpha$
      denominator_2 $\leftarrow \sum$ all entries in row d of $n\_dk + k \cdot \alpha$
      term_2 $\leftarrow \frac{\text{numerator\_2}}{\text{denominator\_2}}$
    Set prob[i] $\leftarrow$ term_1 $\times$ term_2
end for
return $\frac{\text{prob}}{\sum \text{prob}}$

This function is an integral part of the sampling process where the convergence to the true distribution gets closer. Essentially, conditional probability allows Gibbs sampling to efficiently update topic assignments by considering the relative likelihood of different topics for a word within the context of the specific document. The word is then reassigned to the topic with the highest relative probability calculated. (See Equation 2.6)

Algorithm 4 `runGibbs`

Input:

- n_{dk}: Document-topic counts
- n_k: Per-topic word counts
- n_{kw}: Topic-word counts
- docs_assignments: List of word assignments to topics for each document
- n_gibbs: Number of Gibbs sampling iterations
- documents: Collection of documents
- vocabulary: Collection of words

Output:

- Updated n_{dk}: Document-topic counts
- Updated n_k: Per-topic word counts
- Updated n_{kw}: Topic-word counts
- Updated docs_assignments: List of word assignments to topics for each document

Method:

Initialize Gibbs sampling with n_{dk}, n_k, n_{kw}, docs_assignments from `initGibbs`

for t in range(n_gibbs) **do**
    for doc_idx in documents **do**
        for word_idx in doc **do**
            topic ← assign[doc_idx][word_idx]
            position ← index from vocabulary for word
            if position ≠ None **then**
                n_{dk}[doc_idx, topic] ← n_{dk}[doc_idx, topic] − 1
                n_{kw}[topic, position] ← n_{kw}[topic, position] − 1
                n_k[topic] ← n_k[topic] − 1
                prob ← conditionalProbability(position, doc_idx, k)
                // Sample a new topic based on probability distribution
                topic ← sample_topic(prob)
                // Increment counters
                assign[doc_idx][word_idx] ← topic
                n_{kw}[topic, position] ← n_{kw}[topic, position] + 1
                n_{dk}[doc_idx, topic] ← n_{dk}[doc_idx, topic] + 1
                n_k[topic] ← n_k[topic] + 1
            end if
        end for
    end for
end for

The algorithm initially retrieves the current topic assignment for the word and its corresponding position in the vocabulary. This is followed by deducting the counts by 1 in each of the matrices inputted into the function (See Algorithm 4) based on the retrieved topic assignment, index of the word, and the index of the document. After removing the word from its current context, conditional probabilities are calculated with the other topics and the topic number with highest relative probability will be the new topic assignment to the word. The loop is concluded by incrementing counters for word-topic assignments, document-topic assignments, and total word counts per topic are incremented to reflect the new assignment. This continues for a fixed set of iterations and the quality of convergence is tested subsequently using a coherence score at the end.

The next section will delve into the specifics of this rough k-means clustering process and how the centroid vectors, representing potential topics, are inferred in RKMS-LDA.

4.3 RKMS-LDA

The rough k-means clustering process yields centroid vectors as well as lower and upper approximation information for each document for the number of topics based on a TF-IDF matrix as a data matrix given as input(See Algorithm 1). This TF-IDF matrix reflects the numerical representation of a word's importance pertaining to a specific document. The centroids generated from the output of Algorithm 1 can be interpreted as virtual representations of the key topics or themes within each cluster. By analyzing the centroid vectors, we gain a sense of how strongly each word subscribes to a particular cluster based on its presence and weight within the centroid's TF-IDF vector. Next, based on the centroid information, we exploit each of the TF-IDF matrix column where each column represents a word. For each word, we have k values corresponding to the k topics identified by rough k-means. Furthermore, we establish a probability distribution for words within each topic, we normalize these k values by dividing them by the sum of the entire column. This effectively converts the column into a set of probabilities for each word corresponding to each topic. This probability distribution becomes the foundation for topic assignment. Instead of random initialization, we leverage these topic probabilities as weights during sampling. Words are assigned topics based on a weighted sampling process. We've shown experimentally in Chapter 5 that this approach leads to a faster convergence to the true posterior distribution compared to conventional random initializa-

tion used in collapsed Gibbs sampling. The process is expressed in detail in Algorithm 5.

Algorithm 5 `initRoughkmeansGibbs`

Input:
  n_{topics}: Number of topics
  vocab: Vocabulary list
  docs: List of documents
  α, β: Dirichlet priors
Output:
  n_{dk}: Document-topic matrix
  n_k: Words per topic
  n_{kw}: Topic-word matrix
  docs_assignments: List of topic assignments for each document
Method:
Set α and β Dirichlet priors
$k \leftarrow n_{\text{topics}}$ ▷ Set number of topics
$V \leftarrow$ length(vocab) ▷ Set size of vocabulary
$M \leftarrow$ length(docs) ▷ Set number of documents
$n_{kw} \leftarrow \mathbf{0}_{(k,V)}$ ▷ Initialize topic-word matrix with zeros
$n_{dk} \leftarrow \mathbf{0}_{(M,k)}$ ▷ Initialize document-topic matrix with zeros
$n_k \leftarrow \mathbf{0}_k$ ▷ Initialize words per topic with zeros
Initialize empty list docs_assignments
for doc **in** docs **do**
    Initialize empty list assignments
    for word **in** doc **do**
        Sample the topic_index from the probability distribution from the normalized TF-IDF column.
        Assign word the topic corresponding to topic_index
        Append topic_index to assignments
    end for
    Append assignments to docs_assignments
end for
Update n_{kw} ▷ Initial topic-word counts
Update n_{dk} ▷ Initial document-topic counts
Update n_k ▷ Initial per-topic word counts
return n_{dk}, n_k, n_{kw}, docs_assignments

This work utilizes two initialization algorithms, `initRoughkmeansGibbs` (See Algorithm 5) and `initGibbs` (See Algorithm 2), as inputs for a subsequent algorithm called `runGibbs` (See Algorithm 4). The core functionalities of both algorithms are likely identical except for a difference on line number 23 (See Algorithm 5). We get our centroid vectors by feeding the corpus-specific TF-IDF matrix to `RoughKMeansClustering` (See Algorithm 1). This makes the initialization steps differ slightly at this specific point only.

Subsequently, the key benefit here lies in their role as a better head-start for `runGibbs`. By providing a well-initialized starting point through either `initRoughkmeansGibbs` or `initGibbs`, the sampling process within runGibbs is expected to proceed more efficiently. In simpler terms, these initialization algorithms put `runGibbs` in a more advantageous position for the sampling stage, leading to potentially better results. By already leveraging information from centroids, our topic assignments are better placed than before and converge faster, thus yielding efficiency. The next chapter delves into how this theoretical analysis is backed by experimentation and results.

Chapter 5

Experimentation and Results

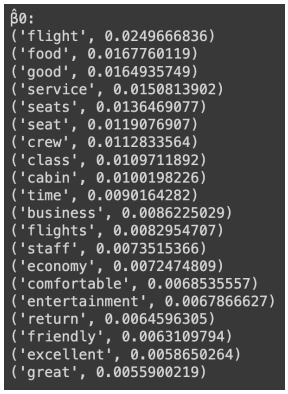
The proposed algorithm RKMS-LDA has been put to experimentation and is compared with the regular collapsed Gibbs Sampling. This evaluation involved five diverse datasets spanning various domains. To ensure a fair comparison, each algorithm was consistently run thrice on each dataset for 100 iterations. The primary focus of the evaluation was to analyze the convergence speed of each algorithm towards achieving the maximum coherence value. Convergence speed is a crucial factor in topic modeling, as faster convergence indicates a more efficient algorithm. The experiments were conducted in a standard Google Colab environment utilizing Python 3, backed by Google Compute Engine. While the specific hardware details (CPU/GPU) might not be critical for this initial evaluation, future explorations could investigate the impact of different computing resources on the performance of RKMS-LDA. Additionally, RKMS-LDA leverages Rough k-means clustering to obtain the centroid vectors of the underlying text corpus [19]. It is to be noted that our experiments revealed minimal discrepancies between the topic-word and document-topic distributions produced by both regular LDA and RKMS-LDA. Therefore, the visualizations presented throughout the experimentation phase generally depict common characteristics of both distributions observed across both algorithms, rather than pertaining to the output of a single method. Finally, the crux of the evaluation focuses on the number of iterations required for RKMS-LDA to achieve convergence compared to LDA, with illustrations provided to support these observations.

5.1 Skytrax User Reviews Dataset

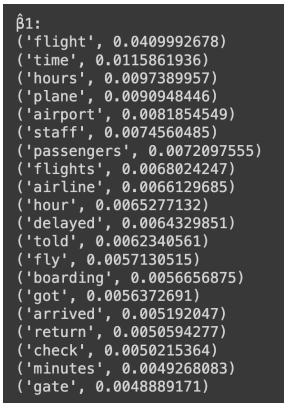
The experiment utilized a dataset of user airline reviews sourced from Skytrax[20]. This dataset can be found on Kaggle as the "Skytrax Airline

Review" dataset or retrieved from the "Skytrax User Reviews Dataset" available on Github. This dataset encompasses a total of 6,266 user reviews. This dataset contains users providing feedback on various aspects of their travel experience. These reviews can pertain to the airline itself, the airport facilities, the service quality within lounges, or the in-flight experience.

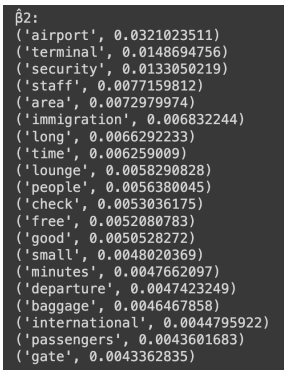
In the final iteration, document-topic and topic-word assignments are as follows:



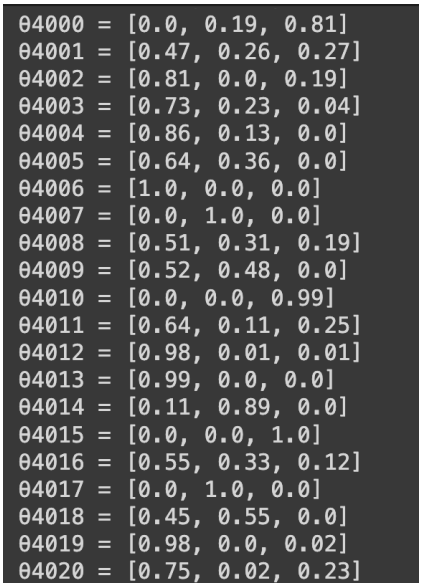
(a) Topic 1



(b) Topic 2



(c) Topic 3



(d) Document-Topic distribution

- **Topic-0**: This topic discusses in-flight experiences and service quality, focusing on comfort, food, and staff performance.

- **Topic-1**: This topic discusses flight schedules, delays, and the boarding process, highlighting issues with punctuality and communication with respect to an airline.
- **Topic-2**: This topic focuses on the airport experience, including terminal facilities, security, and immigration services, narrowing reviews on airports.

	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
LDA	0.30698	0.30449	0.30729	0.32226	0.36087	0.37902	0.39885	0.41154	0.42564	0.42369	0.48496	0.45885	0.49617	0.49524	0.48271	0.48539	0.48209	0.47414	0.48409	0.47855
RKMS-LDA	0.33854	0.47194	0.49682	0.48871	0.49565	0.49565	0.48471	0.47473	0.47473	0.47033	0.46877	0.47550	0.48793	0.47814	0.47394	0.47488	0.47506	0.48932	0.48349	0.47700
LDA	0.30698	0.30601	0.32980	0.33833	0.35370	0.37286	0.39853	0.43915	0.48394	0.48869	0.48417	0.47808	0.48015	0.49179	0.48055	0.47986	0.48306	0.48209	0.47623	0.48197
RKMS-LDA	0.34594	0.46223	0.49214	0.49965	0.49565	0.50387	0.48468	0.46942	0.47473	0.48468	0.46942	0.46942	0.46942	0.48280	0.47033	0.46877	0.48256	0.47970	0.47275	0.48390
LDA	0.30698	0.30698	0.32181	0.33246	0.36822	0.37464	0.39465	0.40973	0.41074	0.48049	0.50105	0.48734	0.49156	0.50767	0.49246	0.48432	0.48533	0.48445	0.48209	0.48178
RKMS-LDA	0.34524	0.47601	0.49682	0.49496	0.49496	0.49565	0.49565	0.48785	0.49024	0.47506	0.48297	0.48760	0.48893	0.47970	0.48197	0.47621	0.48104	0.47814	0.48453	0.48297

Table 5.1: Comparison of Coherence Scores Between LDA and RKMS-LDA for the Skytrax User Reviews Dataset

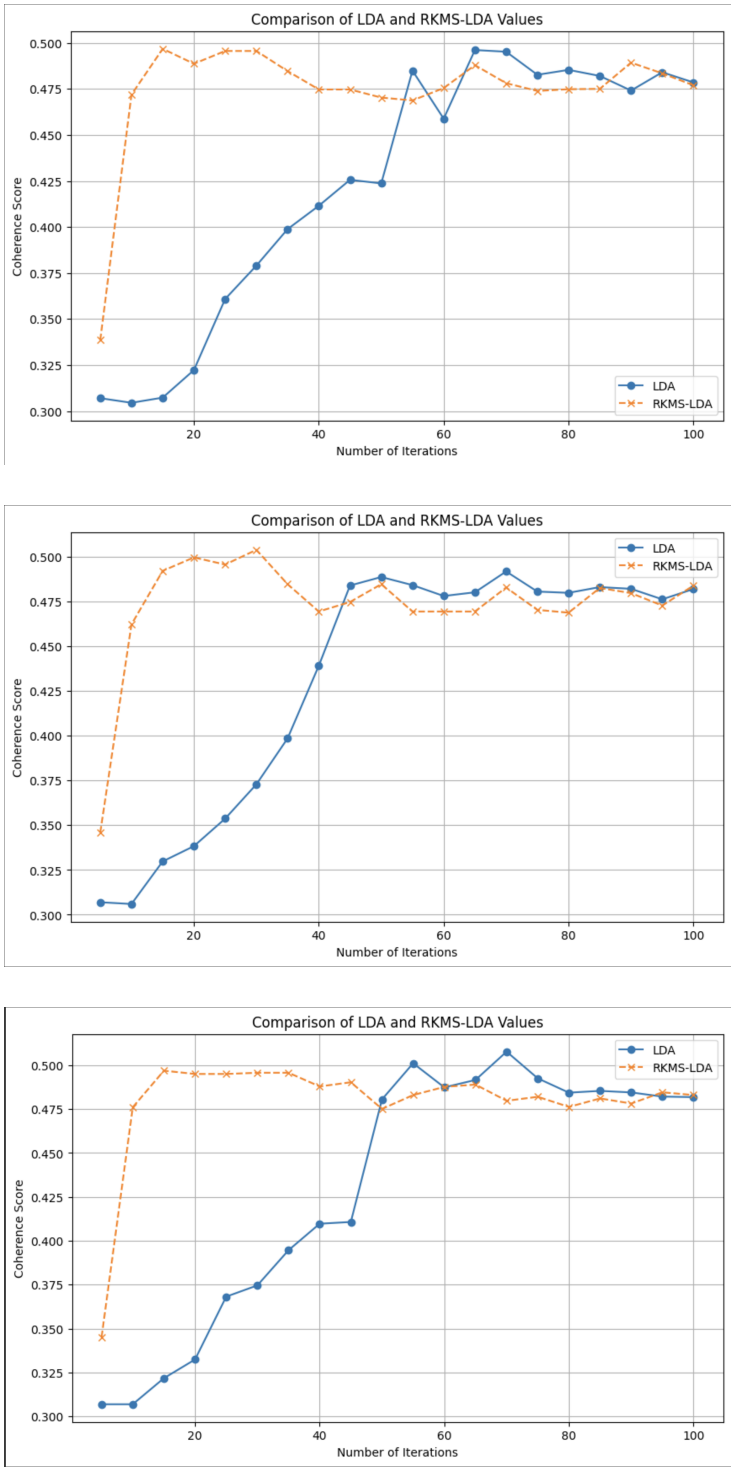


Figure 5.2: Performance Comparison of LDA and RKMS-LDA for the Skytrax User Reviews Dataset

In the first run, RKMS-LDA starts with a higher coherence score (0.3385) compared to LDA (0.3070) at iteration 5. RKMS-LDA achieves its highest coherence score (0.4968) by iteration 15, maintaining high scores throughout, while LDA gradually increases its coherence, peaking at 0.4962 at iteration 65. This indicates RKMS-LDA's rapid convergence and stable performance.

In the second run, RKMS-LDA again begins with a higher coherence score (0.3459) than LDA (0.3070) at iteration 5. RKMS-LDA reaches its peak coherence score (0.5039) by iteration 30 and remains consistent, whereas LDA shows a slower but steady improvement, reaching its peak (0.4918) at iteration 70. RKMS-LDA consistently outperforms LDA in the earlier iterations.

In the third run, the initial coherence scores are similar to the previous runs, with RKMS-LDA at 0.3452 and LDA at 0.3070. RKMS-LDA peaks early with a coherence score of 0.4968 at iteration 15, while LDA peaks later at 0.5077 at iteration 70. Although LDA eventually surpasses RKMS-LDA in coherence score, RKMS-LDA demonstrates faster convergence.

Across all three runs, RKMS-LDA consistently shows a higher initial coherence score and faster convergence compared to LDA. RKMS-LDA reaches its peak coherence scores within the first 30 iterations, whereas LDA requires significantly more iterations to achieve comparable or slightly higher coherence scores.

5.2 BBC News Dataset

This dataset is a collection of 735 news articles from the BBC news website[21]. The dataset is labeled into five categories: entertainment, politics, business, tech, and sports. This makes it a ground truth to reference from for evaluating our Gibbs Sampling used to infer the LDA model.

In the final iteration, the document-topic and topic-word distribution are as follows:

```
β̂0:
('year', 0.0083505723)
('film', 0.0082816163)
('music', 0.0080057923)
('years', 0.0058681561)
('number', 0.0057302441)
('song', 0.0055233761)
('award', 0.0052475521)
('british', 0.004695904)
('set', 0.004144256)
('album', 0.0040753)

β̂1:
('government', 0.0099253698)
('people', 0.008674085)
('labour', 0.0067971578)
('party', 0.0056352505)
('minister', 0.0049202306)
('election', 0.0048755419)
('public', 0.0047414756)
('blair', 0.0042052107)
('brown', 0.0034455021)
('howard', 0.0029539259)
```

(a) Entertainment and Politics

```
β̂2:
('people', 0.0102083333)
('technology', 0.0081784188)
('users', 0.0052403846)
('digital', 0.0050267094)
('mobile', 0.0045459402)
('data', 0.0041720085)
('broadband', 0.0039583333)
('computer', 0.0038514957)
('phone', 0.0038514957)
('players', 0.0036378205)
β̂3:
('year', 0.0088263578)
('company', 0.0082828114)
('market', 0.0071957185)
('sales', 0.0046452314)
('growth', 0.004101685)
('bank', 0.0039762512)
('firm', 0.0039762512)
('firms', 0.0039344399)
('prices', 0.0035581386)
('business', 0.003307271)
```

(b) Technology and Business

```
β̂4:
('game', 0.0111063674)
('games', 0.0090326909)
('play', 0.0070809954)
('time', 0.0070809954)
('think', 0.0050683093)
('club', 0.0049463284)
('going', 0.004275433)
('players', 0.0042144425)
('want', 0.0040924616)
('way', 0.0040924616)
```

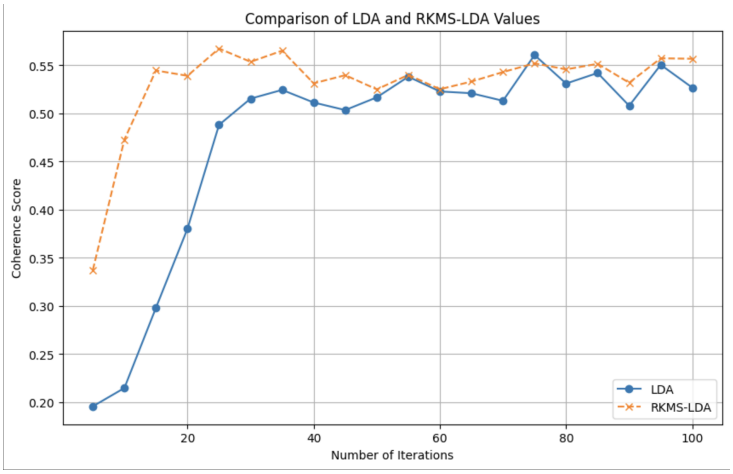
(c) Sports

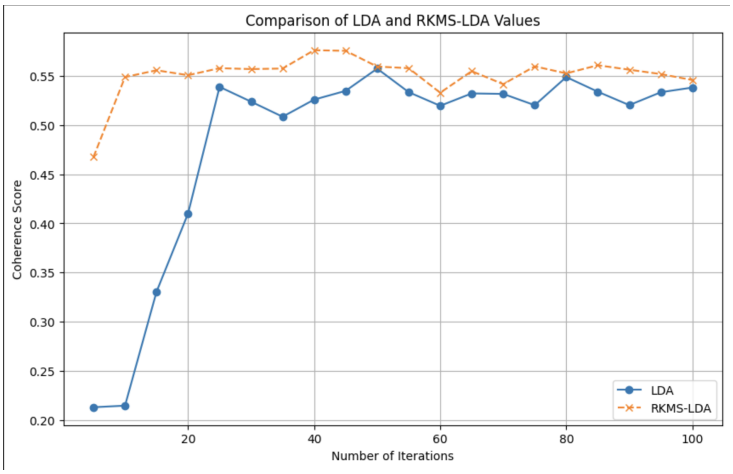
```
θ0 = [0.47, 0.01, 0.01, 0.01, 0.52]
θ1 = [0.0, 0.35, 0.44, 0.21, 0.0]
θ2 = [0.0, 0.0, 0.98, 0.0, 0.0]
θ3 = [0.03, 0.27, 0.0, 0.7, 0.0]
θ4 = [0.0, 0.0, 0.36, 0.0, 0.63]
θ5 = [0.0, 0.4, 0.06, 0.0, 0.53]
θ6 = [0.02, 0.93, 0.0, 0.0, 0.05]
θ7 = [0.1, 0.85, 0.0, 0.0, 0.05]
θ8 = [0.72, 0.01, 0.0, 0.0, 0.26]
θ9 = [0.0, 0.02, 0.06, 0.92, 0.0]
θ10 = [0.0, 0.13, 0.0, 0.84, 0.03]
θ11 = [0.0, 0.0, 0.79, 0.19, 0.03]
θ12 = [0.16, 0.74, 0.01, 0.08, 0.01]
θ13 = [0.0, 0.12, 0.53, 0.35, 0.0]
θ14 = [0.55, 0.0, 0.35, 0.0, 0.1]
θ15 = [0.0, 0.31, 0.12, 0.0, 0.57]
θ16 = [0.09, 0.85, 0.0, 0.0, 0.06]
θ17 = [0.0, 0.0, 0.78, 0.11, 0.11]
θ18 = [1.0, 0.0, 0.0, 0.0, 0.0]
θ19 = [0.3, 0.58, 0.0, 0.12, 0.0]
θ20 = [0.0, 0.53, 0.0, 0.47, 0.0]
```

(d) Document-Topic distribution

	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
LDA	0.19579	0.21481	0.29812	0.38036	0.48817	0.51543	0.52449	0.51139	0.50354	0.51675	0.53819	0.52288	0.52102	0.51317	0.56072	0.53110	0.54207	0.50802	0.55056	0.52684
RKMS-LDA	0.33735	0.47287	0.54466	0.53914	0.56747	0.55365	0.56520	0.53132	0.53979	0.52505	0.54008	0.52518	0.53317	0.54302	0.55210	0.54579	0.55160	0.53193	0.55731	0.55685
LDA	0.213	0.21463	0.33052	0.41005	0.53881	0.52377	0.50845	0.52587	0.53477	0.55739	0.53345	0.51957	0.53218	0.53167	0.52019	0.54879	0.53391	0.52026	0.53336	0.53814
RKMS-LDA	0.4675	0.54881	0.5557	0.55069	0.55784	0.5569	0.55749	0.576109	0.57547	0.55945	0.55784	0.53287	0.55481	0.54158	0.5597	0.55253	0.56082	0.55629	0.55183	0.54577
LDA	0.20574	0.22857	0.27487	0.3542	0.43001	0.43407	0.46143	0.51316	0.52484	0.54018	0.53473	0.5381	0.53137	0.54477	0.54649	0.53636	0.58551	0.55386	0.55507	0.56468
RKMS-LDA	0.39143	0.44202	0.44406	0.47815	0.4659	0.46987	0.48355	0.47787	0.48819	0.51193	0.48358	0.50012	0.50047	0.52742	0.53385	0.49808	0.52407	0.52387	0.53442	0.5387

Table 5.2: Comparison of Coherence Scores Between LDA and RKMS-LDA for the BBC News Dataset.





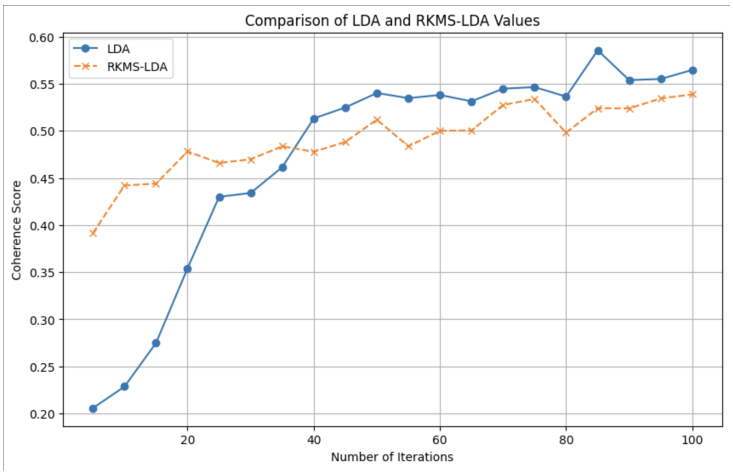


Figure 5.4: Performance Comparison of LDA and RKMS-LDA for the BBC News Dataset

In the first run, RKMS-LDA starts with a significantly higher coherence score (0.33735) compared to LDA (0.19579) at iteration 5. RKMS-LDA quickly peaks at 0.56747 by iteration 25, maintaining high performance and only slightly declining towards the later iterations. In contrast, LDA shows a gradual improvement, peaking at 0.56072 at iteration 65. RKMS-LDA consistently outperforms LDA in the earlier stages, demonstrating faster convergence.

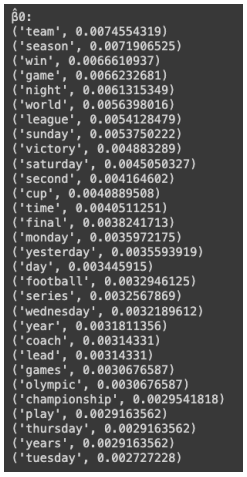
In the second run, RKMS-LDA again begins with a higher coherence score (0.4675) than LDA (0.213) at iteration 5. RKMS-LDA reaches its peak coherence score (0.576109) by iteration 35 and remains relatively stable. LDA's coherence score improves steadily, peaking at 0.55739 by iteration 45, but does not surpass RKMS-LDA. This run highlights RKMS-LDA's efficiency in achieving high coherence quickly and sustaining it throughout.

In the third run, the initial coherence scores show RKMS-LDA at 0.39143 and LDA at 0.20574. RKMS-LDA peaks at 0.5387 by iteration 95, demonstrating a more stable and quicker convergence pattern. LDA gradually improves, reaching its peak of 0.58551 at iteration 85. Although LDA eventually achieves a higher score, RKMS-LDA consistently shows better performance in the early iterations.

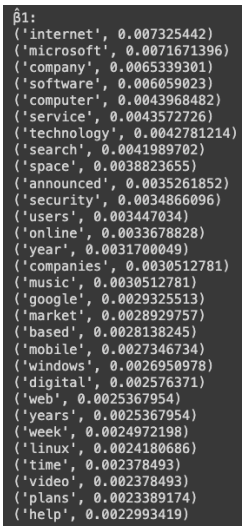
RKMS-LDA achieves its peak coherence scores within the first 35 iterations, while LDA requires significantly more iterations to reach comparable scores.

5.3 AG News Dataset

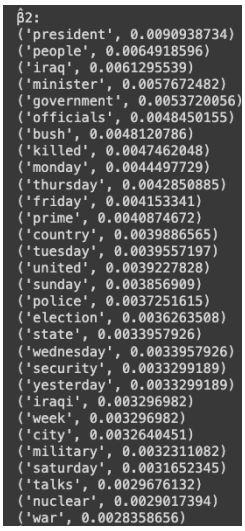
The AG is a massive collection of over 1 million news articles. For more than a year, ComeToMyHead has compiled news articles from over 2000 news sources. This dataset consists of 7600 news articles and focuses on the four most prominent categories within the original corpus: World, Sports, Business, and Science & Technology[22]. In the final iteration, the document-topic and topic-word distribution are as follows:



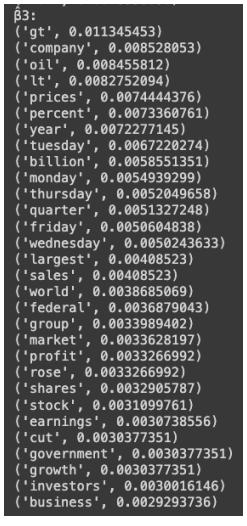
(a) Sports



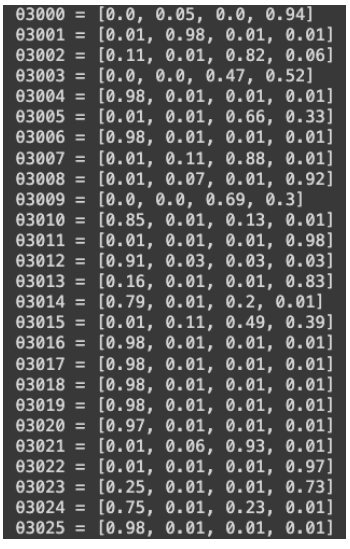
(b) Technology



(c) World



(d) Business



(e) Document-Topic distribution

	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
LDA	0.20464	0.21646	0.21466	0.29779	0.35563	0.42685	0.43225	0.43151	0.43869	0.46376	0.45537	0.47108	0.50115	0.46723	0.50121	0.50522	0.47147	0.45904	0.46408	0.47045
RKMS-LDA	0.24268	0.29627	0.33439	0.39124	0.40544	0.42846	0.43249	0.4706	0.46617	0.45487	0.45804	0.46167	0.4699	0.46489	0.45472	0.4651	0.46459	0.46337	0.46614	0.47094
LDA	0.19905	0.22883	0.25059	0.31147	0.34891	0.3628	0.41413	0.41919	0.41089	0.42663	0.43552	0.43563	0.42961	0.45894	0.4518	0.45064	0.45758	0.48384	0.47975	0.47021
RKMS-LDA	0.266281	0.301278	0.335116	0.349977	0.363551	0.375635	0.422987	0.43116	0.461231	0.463613	0.460315	0.458732	0.464075	0.461963	0.475059	0.457448	0.469375	0.475794	0.477048	0.460607
LDA	0.26439	0.2568	0.24202	0.24959	0.28446	0.28158	0.33116	0.36243	0.3679	0.38176	0.40522	0.4069	0.41241	0.41437	0.41994	0.44546	0.4391	0.44766	0.45425	0.44441
RKMS-LDA	0.29496	0.34222	0.35567	0.38991	0.38261	0.37697	0.40256	0.4005	0.39674	0.43917	0.40759	0.4776	0.46618	0.45967	0.47548	0.49588	0.48122	0.4847	0.49032	0.50518

Table 5.3: Comparison of Coherence Scores Between LDA and RKMS-LDA for the BBC News Dataset.

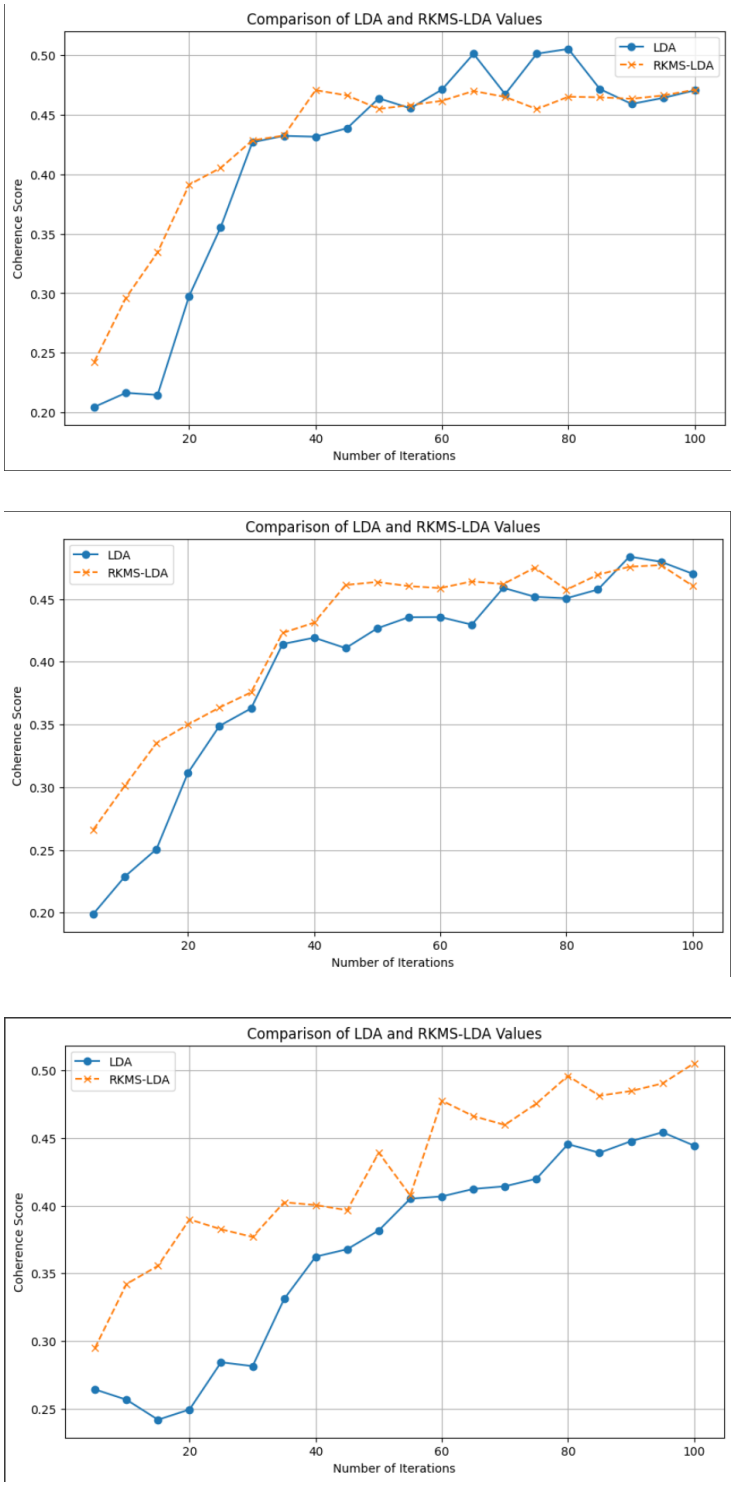


Figure 5.6: Performance Comparison of LDA and RKMS-LDA for the AG News Dataset

In the first run, RKMS-LDA begins with a higher coherence score (0.24268) than LDA (0.20464) at iteration 5. RKMS-LDA continues to increase rapidly, reaching 0.39124 by iteration 20 and peaks at 0.47094 by iteration 95. LDA, on the other hand, shows a slower and steadier increase, peaking at 0.50522 by iteration 80 but generally maintains a lower coherence score compared to RKMS-LDA until the later iterations. This indicates that RKMS-LDA converges faster and maintains a high coherence throughout the iterations.

In the second run, RKMS-LDA again starts with a higher coherence score (0.266281) compared to LDA (0.19905) at iteration 5. RKMS-LDA continues to show a steady increase, reaching 0.475794 by iteration 90, and maintaining its performance thereafter. LDA shows a more gradual improvement, peaking at 0.48384 by iteration 85 but generally lags behind RKMS-LDA throughout the iterations. This run also highlights RKMS-LDA's efficiency in achieving high coherence quickly and sustaining it.

In the third run, RKMS-LDA starts with a higher coherence score (0.29496) than LDA (0.26439) at iteration 5. RKMS-LDA shows a steady increase, reaching a peak coherence of 0.50518 by iteration 95. LDA's coherence score improves more gradually, reaching its peak of 0.45425 at iteration 90. Throughout the iterations, RKMS-LDA consistently maintains higher coherence scores compared to LDA, demonstrating faster convergence and better performance.

RKMS-LDA achieves peak coherence scores within the first 25 to 35 iterations and maintains high coherence levels throughout the iterations while LDA shows a more gradual improvement, often taking significantly more iterations to reach comparable scores.

5.4 Research Articles Dataset

Researchers have access to extensive online archives of scientific research articles. This is a dataset of 1,727 articles, which includes only the abstracts or titles[23]. We determined that using six topics (k=6) yielded the highest coherence when compared to other numbers of topics. In the final iteration, the document-topic and topic-word distribution are as follows:

```
β̂0:
('problem', 0.0119586242)
('algorithm', 0.0117925899)
('time', 0.0085134114)
('function', 0.0071021194)
('model', 0.0067285421)
('distribution', 0.0065625078)
('number', 0.0061889305)
('graph', 0.0056078103)
('paper', 0.0049021643)
('probability', 0.0049021643)

β̂1:
('data', 0.0096053692)
('model', 0.007522168)
('control', 0.0067975762)
('network', 0.0064805674)
('based', 0.0056201147)
('paper', 0.0050313839)
('approach', 0.0048502359)
('systems', 0.0045332271)
('models', 0.0042162182)
('time', 0.0040803572)
```

(a) Topic 1 and Topic 2

```
β̂2:
('data', 0.0140047751)
('learning', 0.0127030391)
('method', 0.0108298091)
('proposed', 0.0089565792)
('model', 0.0088930798)
('neural', 0.0080993383)
('networks', 0.00794059)
('performance', 0.0074008458)
('network', 0.0073690962)
('methods', 0.0072420975)
β̂3:
('equations', 0.0072285305)
('prove', 0.0071693288)
('space', 0.0066365133)
('paper', 0.0064589081)
('equation', 0.0056892857)
('solutions', 0.0055708822)
('case', 0.0052156719)
('theory', 0.0050972684)
('boundary', 0.0046828564)
('group', 0.004327646)
```

(b) Topic 3 and Topic 4

```
β̂4:
('gas', 0.0065002911)
('energy', 0.005753989)
('mass', 0.005753989)
('data', 0.0049330567)
('galaxies', 0.0047837963)
('stars', 0.0045599057)
('clusters', 0.0042613848)
('observations', 0.0042613848)
('model', 0.003962864)
('rate', 0.0038882338)

β̂5:
('model', 0.0111783433)
('phase', 0.008627375)
('quantum', 0.0084743169)
('magnetic', 0.0075559682)
('field', 0.0074539295)
('energy', 0.0054641742)
('order', 0.0053111161)
('state', 0.0052600967)
('dynamics', 0.0052090774)
('states', 0.004596845)
```

(c) Topic 5 and Topic 6

```
θ0 = [0.01, 0.01, 0.01, 0.01, 0.01, 0.97]
θ1 = [0.01, 0.01, 0.01, 0.01, 0.57, 0.41]
θ2 = [0.06, 0.0, 0.83, 0.0, 0.1, 0.0]
θ3 = [0.0, 0.38, 0.59, 0.0, 0.0, 0.02]
θ4 = [0.24, 0.0, 0.75, 0.0, 0.0, 0.0]
θ5 = [0.0, 0.0, 0.0, 0.0, 0.09, 0.91]
θ6 = [0.09, 0.0, 0.74, 0.0, 0.17, 0.0]
θ7 = [0.0, 0.87, 0.01, 0.0, 0.0, 0.11]
θ8 = [0.24, 0.07, 0.45, 0.06, 0.0, 0.18]
θ9 = [0.68, 0.16, 0.16, 0.0, 0.0, 0.0]
θ10 = [0.84, 0.0, 0.0, 0.15, 0.0, 0.0]
θ11 = [0.67, 0.0, 0.29, 0.0, 0.04, 0.0]
θ12 = [0.72, 0.0, 0.27, 0.0, 0.0, 0.0]
θ13 = [0.06, 0.01, 0.01, 0.92, 0.01, 0.01]
θ14 = [0.09, 0.0, 0.0, 0.74, 0.0, 0.17]
θ15 = [0.0, 0.34, 0.62, 0.0, 0.03, 0.0]
θ16 = [0.01, 0.01, 0.01, 0.97, 0.01, 0.01]
θ17 = [0.0, 0.0, 0.11, 0.0, 0.66, 0.23]
θ18 = [0.0, 0.08, 0.82, 0.0, 0.0, 0.1]
θ19 = [0.19, 0.0, 0.27, 0.54, 0.0, 0.0]
θ20 = [0.0, 0.52, 0.48, 0.0, 0.0, 0.0]
θ21 = [0.0, 0.0, 0.0, 0.0, 0.35, 0.64]
θ22 = [0.0, 0.0, 0.14, 0.0, 0.79, 0.05]
θ23 = [0.02, 0.32, 0.17, 0.47, 0.02, 0.02]
θ24 = [0.16, 0.7, 0.01, 0.0, 0.0, 0.14]
θ25 = [0.0, 0.24, 0.75, 0.0, 0.0, 0.0]
```

(d) Document-Topic distribution

- **Topic 0**: This topic focuses on algorithms and computational methods, including optimization, approximation, and statistical methods.

- **Topic 1**: This topic revolves around control systems and networks, with an emphasis on modeling and analyzing complex systems.
- **Topic 2**: This topic is centered on machine learning and deep learning methods, focusing on neural networks, algorithms, and training models.
- **Topic 3**: This topic deals with theoretical aspects of equations, space, and mathematical proofs.
- **Topic 4**: This topic focuses on cosmological phenomena such as gas dynamics, galaxy formations, and observational studies.
- **Topic 5**: This topic revolves around quantum mechanics and condensed matter physics. It covers phases, quantum states, magnetic fields, and interactions between particles.

This dataset lacks pre-assigned categories or labels. The best way to infer the best model is to use the coherence score and subject the generated topics to human perception.

	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
LDA	0.30555	0.32068	0.36393	0.40071	0.46040	0.45596	0.47458	0.47135	0.47304	0.47465	0.46027	0.47570	0.47181	0.45950	0.46819	0.47038	0.47050	0.46161	0.46542	0.46337
RKMS-LDA	0.41875	0.42901	0.44251	0.46227	0.45092	0.46932	0.46162	0.46300	0.45564	0.46432	0.45895	0.48230	0.47974	0.47290	0.44168	0.46170	0.46310	0.47817	0.46255	0.46181
LDA	0.30391	0.30730	0.40620	0.45770	0.47326	0.49027	0.44865	0.45989	0.45608	0.44772	0.44847	0.44732	0.45142	0.44146	0.45929	0.47233	0.49578	0.48199	0.46478	0.48196
RKMS-LDA	0.41300	0.44004	0.42050	0.44218	0.41579	0.42965	0.42369	0.42385	0.42742	0.45773	0.44748	0.45128	0.45641	0.45105	0.46918	0.46445	0.47996	0.45664	0.46862	0.46020
LDA	0.29249	0.32785	0.38480	0.40621	0.43026	0.41714	0.42751	0.44480	0.45868	0.45081	0.45010	0.45575	0.47761	0.49081	0.47910	0.49490	0.48154	0.48036	0.48765	0.47760
RKMS-LDA	0.43311	0.44557	0.46950	0.48018	0.47140	0.44823	0.47155	0.48124	0.47196	0.46363	0.46001	0.46205	0.46011	0.46603	0.46863	0.47086	0.46407	0.45956	0.45943	0.48075

Table 5.4: Comparison of Coherence Scores Between LDA and RKMS-LDA for the Research Articles Dataset.

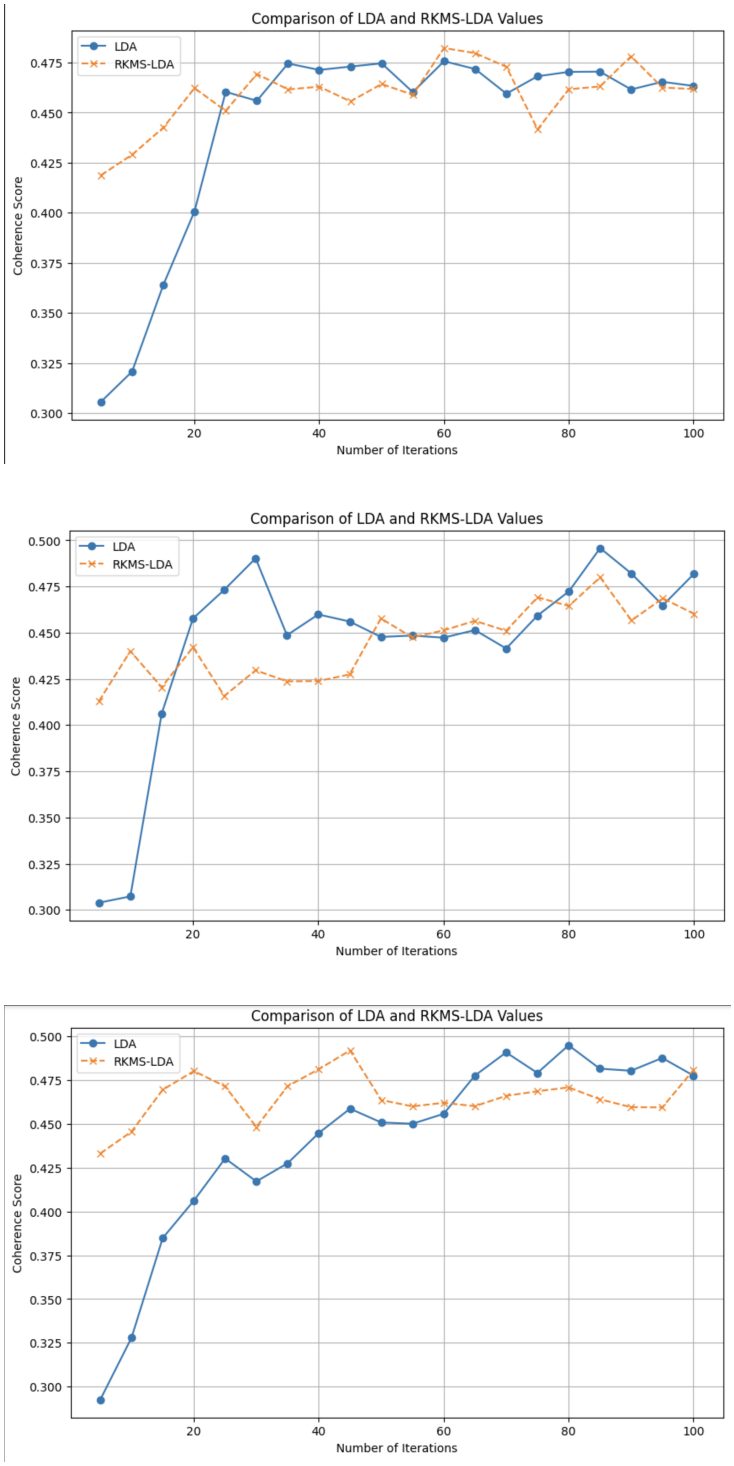


Figure 5.8: Performance Comparison of LDA and RKMS-LDA for the Research Articles Dataset

In the first run, RKMS-LDA begins with a significantly higher coherence score (0.41875) compared to LDA (0.30555) at iteration 5. RKMS-LDA reaches its peak coherence of 0.4823 by iteration 55, and generally maintains a high level of coherence. LDA, while showing a steady increase, peaks at 0.4757 by iteration 55 but then slightly declines, ending at 0.46337 by iteration 100. Throughout this run, RKMS-LDA consistently maintains higher coherence scores compared to LDA, demonstrating faster convergence and better stability.

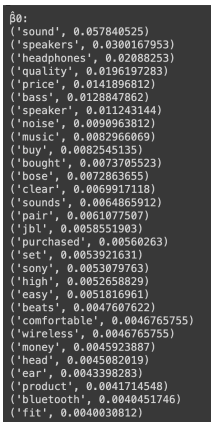
In the second run, RKMS-LDA starts with a higher coherence score (0.413) compared to LDA (0.30391) at iteration 5. RKMS-LDA reaches a score of 0.45773 by iteration 50 and continuing to maintain high coherence with minor fluctuations. It peaks at 0.47996 by iteration 85. LDA, on the other hand, improves more gradually, reaching its peak coherence of 0.49578 at iteration 85 but then slightly declining. Again, RKMS-LDA shows faster convergence and more stable performance compared to LDA.

In the third run, RKMS-LDA begins with a higher coherence score (0.43311) compared to LDA (0.29249) at iteration 5. RKMS-LDA continues to increase steadily, reaching 0.48124 by iteration 40. LDA shows a gradual increase, peaking at 0.4949 by iteration 80, but fluctuates more towards the end.

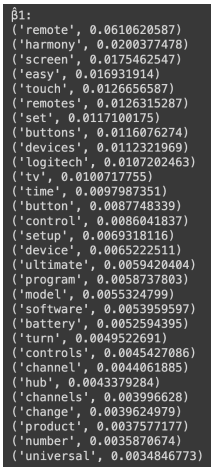
RKMS-LDA reaches peak coherence scores within the first 25 to 35 iterations, and maintains high coherence levels throughout the iterations. In contrast, LDA shows a more gradual improvement, often peaking later and showing more fluctuations towards the end.

5.5 Amazon and Best Buy Electronics Reviews Dataset

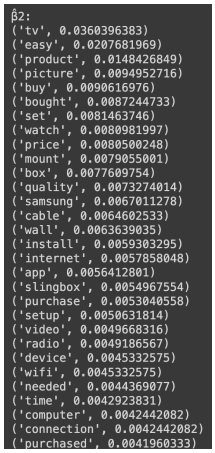
This dataset comprises over 7,000 online reviews for 50 electronic products sourced from websites like Amazon and Best Buy, provided by Datafiniti's Product Database. Each review includes details such as the review date, source, rating, title, reviewer metadata, and more[24]. After topic modeling, k=5 topics were found to yield higher coherence scores, allowing for better interpretability of the topics. In the final iteration, the document-topic and topic-word distribution are as follows:



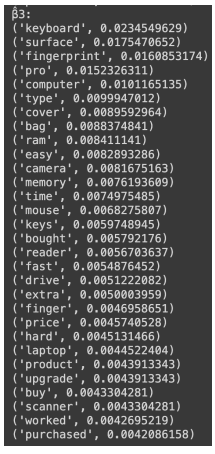
(a) Topic 1



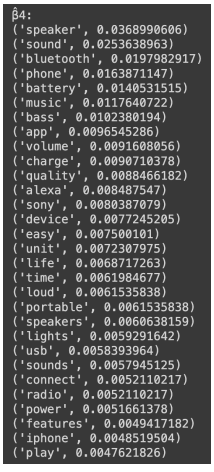
(b) Topic 2



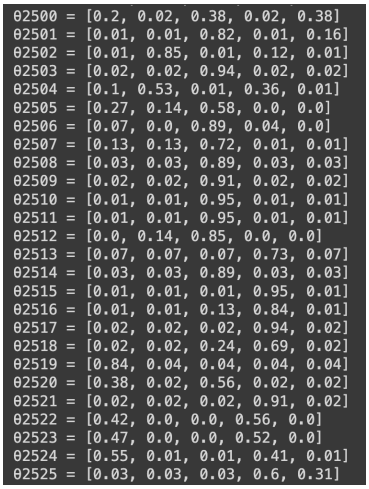
(c) Topic 3



(d) Topic 4



(e) Topic 5



(f) Document-Topic distribution

- **Topic 0**: This topic focuses on the reviews associated with the sound quality of audio equipment, including speakers, headphones, and other sound devices. This is quite evident in words like Bose, JBL, and Sony.
- **Topic 1**: This topic revolves around remote controls, particularly those by Logitech Harmony. It covers aspects like the ease of setup, functionality of touch screens and buttons, device compatibility, and user experience.
- **Topic 2**: This topic pertains to televisions and related devices, including smart TVs, streaming boxes, and mounts.
- **Topic 3**: This topic is about computer peripherals and accessories, focusing on keyboards, fingerprint readers, and other input devices. It could also include components like RAM, hard drives, and memory.
- **Topic 4**: This topic deals with portable speakers, especially those with Bluetooth connectivity. Products like Amazon Alexa, Sony speakers, and other Bluetooth-enabled devices are frequently mentioned, with emphasis on their portability and audio performance.

	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
LDA	0.37488	0.507575	0.57405	0.57899	0.56185	0.56148	0.55640	0.53328	0.51899	0.53514	0.53387	0.52424	0.53721	0.54342	0.52773	0.53317	0.53194	0.51889	0.54067	0.53714
RKMS-LDA	0.46083	0.55849	0.57	0.56167	0.57718	0.58183	0.59436	0.57330	0.56920	0.57492	0.56884	0.55791	0.54288	0.56686	0.54268	0.54934	0.54948	0.54645	0.53901	0.55577
LDA	0.38928	0.47763	0.50534	0.52093	0.53983	0.53388	0.54786	0.57644	0.54096	0.54435	0.53534	0.55211	0.57000	0.54281	0.55701	0.54567	0.54567	0.55417	0.54679	0.56855
RKMS-LDA	0.37488	0.50757	0.57405	0.57899	0.56185	0.56148	0.55640	0.53328	0.51894	0.53514	0.53387	0.52424	0.53721	0.54352	0.52773	0.53317	0.53194	0.51899	0.54067	0.53714
LDA	0.37977	0.46957	0.52648	0.54397	0.55003	0.57075	0.57501	0.55283	0.56420	0.55947	0.56337	0.56301	0.55476	0.54839	0.55785	0.56760	0.55564	0.55804	0.54092	0.55307
RKMS-LDA	0.42682	0.51200	0.53918	0.53930	0.53346	0.52461	0.51702	0.53102	0.51887	0.53259	0.52357	0.53666	0.54055	0.54305	0.53045	0.53180	0.54843	0.54921	0.54885	0.55043

Table 5.5: Comparison of Coherence Scores Between LDA and RKMS-LDA for the Amazon and Best Buy Electronics Reviews Dataset.

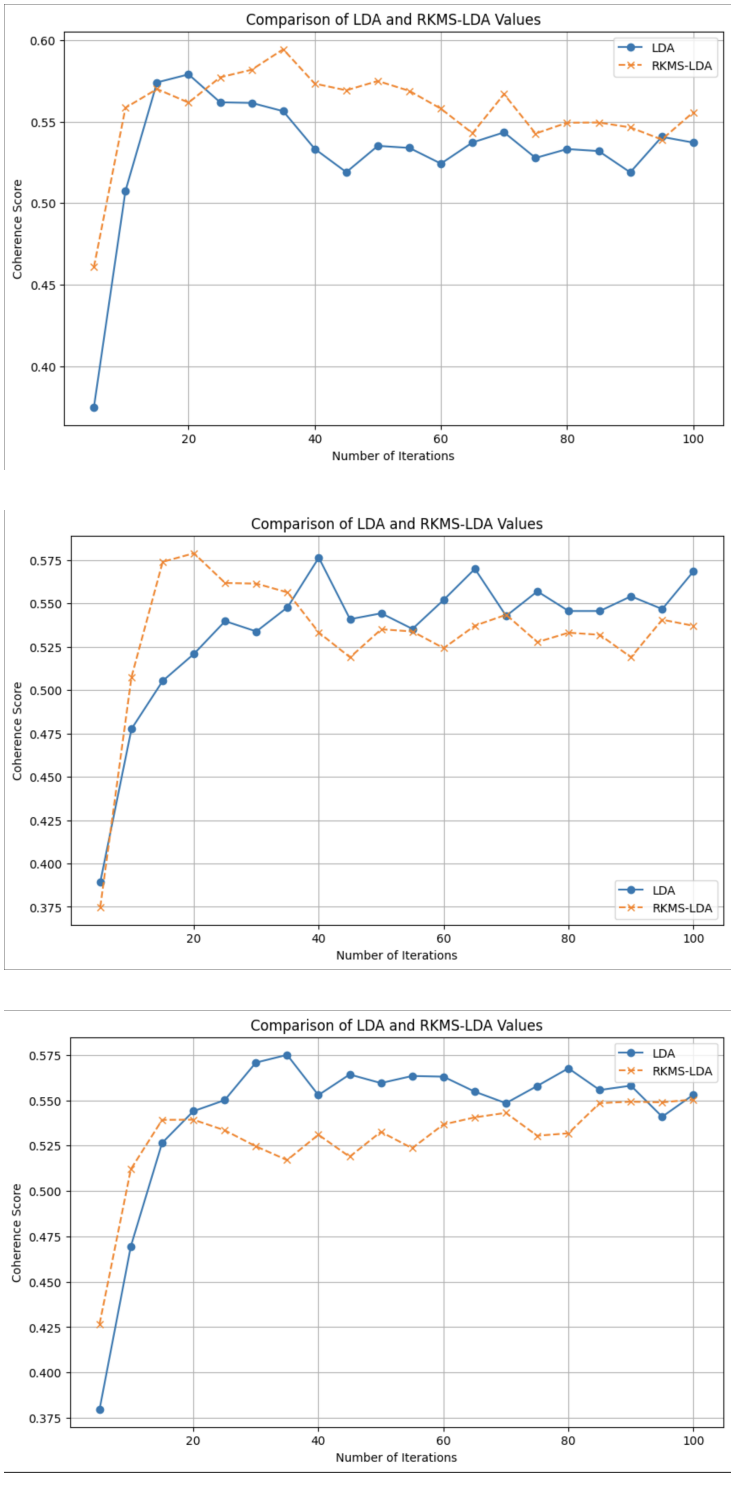


Figure 5.10: Performance Comparison of LDA and RKMS-LDA for the Amazon and Best Buy Electronics Reviews Dataset

In the first run, RKMS-LDA starts with a higher coherence score (0.46083) than LDA (0.37488) at iteration 5. RKMS-LDA quickly improves, reaching 0.59436 by iteration 35, and stays relatively high with some ups and downs. LDA steadily increases, peaking at 0.57899 by iteration 15, but then slowly drops, ending at 0.53714 by iteration 100. Overall, RKMS-LDA performs better and stabilizes faster than LDA.

In the second run, RKMS-LDA begins with a lower coherence score (0.37488) compared to LDA (0.38928) at iteration 5. However, by iteration 10, RKMS-LDA surpasses LDA with a score of 0.50757. RKMS-LDA keeps improving, reaching 0.57405 by iteration 25, and stays high with some fluctuations. LDA improves steadily, peaking at 0.570 by iteration 60 but then slightly drops, ending at 0.53714 by iteration 100. RKMS-LDA shows faster and steadier improvement than LDA.

In the third run, RKMS-LDA starts with a higher coherence score (0.42682) compared to LDA (0.37977) at iteration 5. RKMS-LDA keeps improving, reaching 0.5393 by iteration 20, and stays high with minor fluctuations. LDA steadily increases, peaking at 0.57501 by iteration 30, but then gradually drops, ending at 0.55043 by iteration 100. Again, RKMS-LDA shows faster improvement and higher stability compared to LDA.

Across all three runs, RKMS-LDA consistently starts with higher or comparable initial coherence scores and demonstrates faster convergence compared to LDA. RKMS-LDA reaches peak coherence scores more quickly, typically within the first 20 to 30 iterations, and maintains high coherence levels throughout the iterations.

The algorithm RKMS-LDA and LDA, having been put through comparison many times, often show that RKMS-LDA has a better head start than regular Gibbs sampling. While randomization might occasionally outperform the use of the rough k-means preprocessor (See Figure 5.10), the preprocessor generally provides a calculated advantage by reducing the number of iterations needed to converge the true posterior distribution. We conclude our dissertation in the next chapter, summarizing key findings and outlining promising directions for further research to build upon this work.

Chapter 6

Conclusions and future works

This dissertation focused on enhancing the inference model of Latent Dirichlet Allocation (LDA) by introducing the RKMS-LDA algorithm. This approach leveraged the strengths of soft computing and rough k-means clustering. Instead of random initializing topic-word and document-topic assignments, RKMS-LDA utilizes rough k-means to provide a more informed starting point. This demonstrably improved the overall effectiveness of the LDA model.

Also, our current implementation focuses on sampling from all topic probabilities for each individual word, an interesting extension lies in leveraging document-level information. Rough k-means clustering not only provides centroid vectors but also can identify documents belonging to the lower or upper approximations of each cluster. This suggests a potential optimization for documents within the lower approximation of a cluster, where we could restrict topic sampling only to those associated with that specific cluster.

Looking ahead, exploring a distributed implementation of RKMS-LDA using Apache Spark is also promising. By capitalizing on the power of Resilient Distributed Datasets (RDDs), we can scale this approach to handle massive datasets containing millions of documents.

Secondly, the potential for a streaming version of RKMS-LDA deserves further investigation. Streaming LDA models often struggle to achieve the same level of perplexity and coherence as their batch counterparts[25]. A streaming version of RKMS-LDA could potentially aim at bridging this gap, allowing for real-time topic modeling on large document streams.

In conclusion, RKMS-LDA has attempted to lay the groundwork for advancements in LDA inference using collapsed Gibbs sampling.