Senior Thesis in Mathematics

Gibbs Sampling for LDA and Applications to RAG

Author:
Kyle Torres

Advisor:
Dr. Jo Hardin

Submitted to Pomona College in Partial Fulfillment
of the Degree of Bachelor of Arts

May 5, 2025

Acknowledgements

Thank you to my family and friends for supporting my academic journey and providing an invaluable system of support and kindness.

Abstract

Latent Dirichlet Allocation (LDA) is one of the most popular topic-modeling algorithms in use today. In this work, I describe a method for deriving the posterior distribution used in LDA and create a hybrid model in which I combine LDA with a baseline retrieval-augmented generation (RAG) model. I find that this hybrid model outperforms the baseline RAG model in several areas including accuracy and processing time. These results highlight the potential for LDA to be incorporated in modern RAG-based models as a means of extending their performance from closed-book question answering (QA) tasks to open-domain QA tasks.

Contents

Chapter 1

Introduction

In this paper, I begin by discussing the motivations behind Latent Dirichlet Allocation (LDA), one of the most widely used topic modeling algorithms. To explain its workings, I derive the Gibbs sampling distribution, which provides a method for estimating the relevant posterior distribution and iteratively updating topic distributions in documents. Following this, I review retrieval-augmented generation (RAG), a framework that has the potential to improve large language model responses. To bridge LDA and RAG, I conduct an experiment that incorporates LDA as a pre-processing step to enhance retrieval efficiency in RAG. This experiment has practical implications such as improving model performance and reducing runtime. I conclude by discussing the experiment's results and highlighting potential future directions for optimizing the model's performance.

As the volume of unstructured textual data continues to grow, covering everything from academic publications to social media commentary, there has been an increased demand to organize and interpret this data. Simple keyword-based retrieval methods depend primarily on exact term matches and often overlook more nuanced themes in the data; meanwhile, manual annotation efforts are impractical for large amounts of data. As a result, researchers and practitioners have turned to methods such as *topic modeling*, which can uncover complex latent semantic relationships in textual data.

Topic modeling is an unsupervised machine learning method that identifies latent thematic structure in textual data by organizing documents into "topics." Topic modeling infers both the topic proportions that characterize each

document and the word distributions that define each topic. This dual inference process uncovers the latent thematic structure in the text, effectively mapping high-dimensional text data into a lower-dimensional topic space. Topic modeling can be used in a variety of real-world applications including sentiment analysis, recommender systems, and information retrieval.

Latent Dirichlet Allocation (LDA) is a probabilistic topic modeling algorithm, where documents are represented as mixtures of topics, and topics are characterized by distributions over words. Given a collection of documents, LDA seeks to infer both the underlying topics and the topic distribution for each document. Because the posterior distribution of the latent topic variables in LDA is intractable (impossible to compute directly), approximation techniques such as Gibbs sampling are used to estimate it. Gibbs sampling is a Markov Chain Monte Carlo (MCMC) method that allows for iterative sampling from the joint distribution by conditioning on all other variables except for the current variable. Over time, Gibbs sampling converges to a stable estimate of the posterior distribution over the latent topic assignments, enabling us to efficiently infer each document's topic mixture and each topic's word distribution in LDA.

Chapter 2

Motivating Distributions

In probability, discrete outcomes are often modeled by conjugate distribution pairs. For instance, when flipping a coin n times, the binomial distribution describes the probability of observing a given number of heads, while the beta distribution serves as its conjugate prior, describing one's uncertainty about the coin's bias before any flips occur. Similarly, when drawing multiple balls from an urn containing various colors, the multinomial distribution specifies the expected counts of each color, while the Dirichlet distribution provides a conjugate prior by representing prior probabilities for every color category. These conjugate relationships, beta-binomial for binary outcomes and dirichlet-multinomial for non-binary scenarios, ensure that the posterior distribution remains in the same family as the prior. We will see later in Chapter 3 how Latent Dirichlet Allocation (LDA) uses this dirichlet–multinomial conjugacy to represent each document as a probabilistic mixture over latent topics and each topic as a distribution over words.

2.1 Beta Distribution

The beta distribution is a family of continuous probability distributions whose domain is the interval [0,1]. The distributions are parameterized by α and β, which control the shape of the distribution. The beta distribution will serve as the foundation for the sampling methods I describe in future sections due to its relevance as a conjugate prior for the binomial distribution and its relationship to the Dirichlet distribution. The pdf of the beta distribution is given below:

$$p(\theta \mid \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}, \quad \text{where } 0 \leq \theta \leq 1$$

2.2 Binomial Distribution

The binomial distribution is a discrete probability distribution that describes the number of successes in n independent Bernoulli trials, each of which has a success probability, θ. In a single Bernoulli trial, the random variable takes the value 1 with probability θ and 0 with probability $1-\theta$. Thus, when $n = 1$, the binomial distribution is simply the Bernoulli distribution. The pmf of the binomial distribution is given below:

$$f(x \mid \theta, n) = \binom{n}{x}\theta^{x}(1-\theta)^{n-x}, \quad \text{where } x = 0, 1, 2, \cdots, n$$

where n represents the number of trials and $\theta \in [0, 1]$ represents the success probability for each trial.

2.3 Beta-Binomial Conjugacy

The beta distribution is the conjugate prior distribution for several distributions including the binomial distribution. Conjugate distributions arise when, given a particular likelihood function, the posterior distribution is in the same probability distribution family as the prior distribution. This prior distribution is thus called the conjugate prior, which is represented by the beta distribution in this case. The beta distribution is conjugate to the binomial likelihood.

In general, the posterior distribution can be written as an extension of Bayes Rule, where the numerator is a joint probability:

$$p(\theta \mid x) = \frac{p(x, \theta)}{p(x)} = \frac{p(x|\theta)p(\theta)}{p(x)}$$

where $f(x \mid \theta)$ represents the likelihood function and $p(\theta)$ represents the prior distribution. The posterior is proportional to the joint probability $p(\theta \mid x) \propto p(x|\theta)p(\theta)$.

If we assume the likelihood comes from the binomial distribution and the prior comes from the beta distribution, we can rewrite the posterior as the following:

$$\begin{aligned} p(\theta \mid x) &\propto \binom{n}{x}\theta^x(1-\theta)^{n-x}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1} \\ &= \binom{n}{x}\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{x+\alpha-1}(1-\theta)^{n-x+\beta-1} \end{aligned}$$

It is important to note that we are only considering the numerator of the posterior, which is why we use $\propto$ to represent proportionality instead of equality. If we disregard the constants at the beginning and only consider the terms in the result that contain θ, we notice that the result looks like a beta pdf with new parameters $\alpha' = x + \alpha$ and $\beta' = n - x + \beta$. Thus, the posterior distribution with a binomial likelihood and a beta prior can be characterized as a beta distribution $\theta \mid x \sim Beta(x + \alpha, n - x + \beta)$, so it follows that the beta distribution is a conjugate prior for the binomial distribution.

2.4 Dirichlet Distribution

The Dirichlet distribution is the multivariate generalization of the beta distribution. The Dirichlet distribution is parameterized by $\boldsymbol{\alpha}$ and K where $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_K)$ is a vector, and K represents the number of categories in the probability vector. When $K = 2$, the Dirichlet distribution looks like a beta distribution where $\alpha = \alpha_1$ and $\beta = \alpha_2$. The Dirichlet distribution is relevant for this paper because it is commonly used as a conjugate prior for the multinomial distribution. The pdf of the Dirichlet distribution is given below:

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

2.5 Multinomial Distribution

The multinomial distribution is a generalization of the binomial distribution, as it models the number of successes for each of k categories in n independent trials. The binomial distribution is a special case of the multinomial distribution where $k = 2$. The categorical distribution is a special case where $n = 1$ (multivariate generalization of the Bernoulli distribution). The multinomial distribution comes into play when $k > 2$. The pmf of the multinomial distribution is shown below:

$$p(X_1 = x_1, X_2 = x_2, \cdots, X_k = x_k) = \frac{n!}{x_1!x_2!...x_k!} \prod_{i=1}^{K} \theta_i^{x_i} = \frac{\Gamma(n+1)}{\prod_{i=1}^{K} \Gamma(x_i+1)} \prod_{i=1}^{K} \theta_i^{x_i}$$

2.6 Dirichlet-Multinomial Conjugacy

Like the beta-binomial conjugacy, the dirichlet-multinomial conjugacy involves a conjugate prior-posterior relationship, where the prior and posterior distributions belong to the same family of distributions. However, unlike the beta-binomial conjugacy, the dirichlet-multinomial conjugacy considers generalizations of the beta and binomial distributions where $k > 2$, meaning there are more than two categories.

Let $\theta = (\theta_1, \theta_2, \cdots, \theta_K)$ be the probabilities of a categorical variable with K outcomes. The Dirichlet prior over θ is defined as:

$$p(\theta|\alpha) = \frac{1}{B(\alpha)} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1}$$

where $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_K)$ are the Dirichlet parameters.

Assuming there are n independent trials, where the number of times the k^{th} outcome occurs is x_k, the multinomial likelihood is:

$$p(X|\theta) = \frac{n!}{x_1!x_2!...x_K!} \prod_{i=1}^{K} \theta_i^{x_k}$$

where $\sum_{k=1}^{K} x_k = n$

Now, we can substitute the multinomial likelihood and the Dirichlet prior to compute the posterior:

$$p(\theta|X) \propto \left(\prod_{i=1}^{K} \theta_i^{x_k} \right) \left(\prod_{k=1}^{K} \theta_k^{\alpha_k - 1} \right) = \prod_{k=1}^{K} \theta_k^{(\alpha_k + x_k) - 1}$$

We observe that the posterior looks like a Dirichlet distribution with parameters $\alpha' = (\alpha_1 + x_1, \alpha_2 + x_2, \cdots, \alpha_K + x_K)$, so $P(\theta|X) \sim \text{Dirichlet}(\alpha_1 + x_1, \alpha_2 + x_2, \cdots, \alpha_K + x_K)$, demonstrating the dirichlet-multinomial conjugacy.

Chapter 3

Latent Dirichlet Allocation

3.1 Overview

Introduced in Blei et al. (2003), Latent Dirichlet Allocation (LDA) is a generative probabilistic model of a corpus of text. The main idea of the model is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words. Although LDA provides a generative structure for the corpus, the resulting posterior distribution is intractable. To solve this estimation problem, researchers used an approach called Gibbs sampling to iteratively resamples each word's topic assignment from its conditional distribution given all other topic assignments. Griffiths (2004).

The Gibbs sampler is a Markov Chain Monte Carlo (MCMC) algorithm used to generate samples from a multivariate probability distribution when direct sampling from the joint distribution is impractical. This method works by iteratively sampling from the conditional distributions of individual variables given all of the other variables. Each conditional distribution fixes the other variables and therefore has a density function proportional to the full joint density. For example, we can write the conditional distribution for some variable X_1 given all other variables $X_2, \cdots, X_n$ using Bayes Rule, where the numerator represents the joint distribution:

$$P(X_1|X_2, \cdots, X_n) = \frac{P(X_1, X_2, \cdots, X_n)}{P(X_2, \cdots, X_n)}$$

The denominator is a constant with respect to X_1 because it does not depend

on X_1. Thus, it is included in the normalization constant and is not accounted for in the sampling. Now, the conditional distribution is proportional to the joint distribution:

$$P(X_1|X_2, \cdots, X_n) \propto P(X_1, X_2, \cdots, X_n)$$

Because of this property, we can iteratively sample from conditional distributions, which are often easier to compute than the full joint distribution. By iteratively drawing each variable from its conditional distribution so that each update uses the most recent values of the other variables, Gibbs sampler gradually explores the full joint space. We continue this cycle until convergence, meaning we run enough iterations for the Markov chain to stabilize (and lose memory of the starting values), so our collected samples can accurately reflect the true posterior distribution.

In the context of LDA, the Gibbs sampler is used to approximate the posterior distribution of the topic assignments Z, which are latent variables. The Gibbs sampler approximates this posterior distribution by iteratively sampling the topic assignments from their conditional distributions given the observed words W, the hyperparameters α and β, and the topic assignments of all the other tokens.

3.2 Gibbs Sampling Derivation for LDA

This section is derived from Mukherjee (2016).

Below is a brief list of the notations that will be appear in the derivations:

Notations:
D: # of documents
N: # of words per document (N_d for d^{th} document)
K: # of topics
V: set of unique words/vocabulary size
$\theta_d = \{\theta_{d,1}, \theta_{d,2}, \cdots, \theta_{d,K}\}$: topic distribution for document d
$\phi_k = \{\phi_{k,1}, \phi_{k,2}, \cdots, \phi_{k,V}\}$: word distribution for topic k
$z_{d,n}$: latent topic assignment to n^{th} word of document d (also called z_i where i is a token indexed at a particular word and document)
$w_{d,n}$: n^{th} word of document d (also called w_i)

$Z = \{z_{d,n}\}$: set of latent topic assignments
$W = \{w_{d,n}\}$: set of observed words in the corpus
$\Theta = \{\theta_d\}$: set of document-topic distributions
$\Phi = \{\phi_k\}$: set of topic-word distributions

3.2.1 Joint Distribution

I start by deriving the joint distribution $P(W, Z; \alpha, \beta)$, which defines the probability of both the observed words W and the latent topic assignments Z given parameters α and β. Gibbs sampling iteratively samples topic assignments from conditional probabilities, which are derived from the joint distribution, so calculating the joint distribution is crucial for ensuring the chain eventually converges to the true LDA posterior distribution. The conditional distribution is proportional to the joint distribution:

$$P(z_i = k' | Z_{-i}, W) \propto P(W, Z)$$

The above conditional represents the probability that the topic of i^{th} token is k' given the topic assignments for all other tokens as well as all the words in the corpus. The full conditional probability for the Gibbs sampler is derived throughout this section and provided in equation 19.

The joint distribution can be decomposed into the following:

$$P(W, Z; \alpha, \beta) = P(W|Z)P(Z) = I_1 \times I_2$$

I define I_1 as the conditional probability of the observed words W given the latent topic assignments Z and I_2 as the unconditional probability of the set of latent topic assignments Z. In both I_1 and I_2, we integrate over a set of distributions to find the desired probability:

$$I_1 = P(W|Z) = \int P(W|Z, \Phi)P(\Phi)\, d\Phi \qquad (I_1)$$

$$I_2 = P(Z) = \int P(Z|\Theta)P(\Theta)\, d\Theta \qquad (I_2)$$

I will solve for I_2 first:

$$I_2 = P(Z) = \int P(Z|\Theta)P(\Theta)d\Theta$$

where $\Theta = \{\theta_1, \theta_2, \ldots, \theta_D\}$ is the set of all document-topic distributions in the corpus. For example, θ_1 represents the distribution of topics in document 1, θ_2 represents the distribution of topics in document 2, and so on.

Note that Θ is a D x K matrix where each row $\theta_d = \{\theta_{d,1}, \theta_{d,2}, \ldots, \theta_{d,K}\}$ represents the topic distribution for document d. Each element of the matrix can be represented as the probability that topic k is assigned to a randomly chosen word in document d. For example, $\theta_{d,1}$ represents the probability that topic 1 is assigned to a randomly chosen word in document d.

Next, let's assume a prior distribution such that $\theta_d \sim \text{Dirichlet}(\alpha)$, meaning that for each document d, we draw its topic distribution θ_d from a Dirichlet distribution with some parameter vector $\boldsymbol{\alpha}$. Further, let's assume the document-topic distributions are independent across documents. We can now write the probability of the set of document-topic distributions as the product of probabilities representing each individual document-topic distribution:

$$P(\Theta) = \prod_{d=1}^{D} P(\theta_d|\boldsymbol{\alpha})$$

Next, recall that $\Theta = \{\theta_1, \ldots, \theta_K\} \sim \text{Dirichlet}(\alpha_1, \ldots, \alpha_K)$ has the following PDF in its general form:

$$f(\theta_1, \ldots, \theta_K; \alpha_1, \ldots, \alpha_K) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{K} \theta_i^{\alpha_i - 1}, \quad \text{where } B(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}$$

Because we are integrating the PDF over the simplex in equation I_2 to solve for I_2, the following expression must equal 1 (by definition):

$$\int \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{K} x_i^{\alpha_i - 1} d\Theta = 1 \quad \text{or} \quad \int \prod_{i=1}^{K} x_i^{\alpha_i - 1} d\Theta = B(\boldsymbol{\alpha}) \tag{1}$$

Note that if we move the $\frac{1}{B(\boldsymbol{\alpha})}$ outside the integral, we can rewrite the expression to equal $B(\boldsymbol{\alpha})$.

Using the PDF of the Dirichlet distribution we can expand:

$$P(\Theta) = \prod_{d=1}^{D} P(\theta_d | \boldsymbol{\alpha}) = \prod_{d=1}^{D} \left(\frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{K} \theta_{d,k}^{\alpha - 1} \right) \tag{2}$$

We assume that each hyper-parameter of the K-dimensional Dirichlet distribution is constant, where $\alpha_1 = \cdots = \alpha_K = \alpha$, i.e., the vector $\boldsymbol{\alpha} = \{\alpha_1, \cdots, \alpha_K\} = \{\alpha, \cdots, \alpha\}$. By initializing values of α with equal weight, we assume a non-informative prior.

Recall that:

$$Z = \{z_{d=1,j=1}, \cdots z_{d=1,j=N_1}, \cdots z_{d=D,j=N_D}\}$$

where Z is a vector representing the topic assignments across the entire corpus, $z_{d=1,j=1}$ is the topic assignment for the first word in the first document, $z_{d=1,j=N_1}$ is the topic assignment for the last word in the first document, and $z_{d=D,j=N_D}$ is the topic assignment for the last word in the last document.

We assume that within each document, topic assignments for words $z_{d,j}$ are conditionally independent given the document-topic distribution θ_d. This assumption stems from the bag-of-words assumption in LDA, where the order of words is not taken into account. Instead, each word's topic assignment depends only on θ_d and not on the topic assignments of the other words in the document. We also assume that $z_{d,j}|\theta_d \sim Cat(\theta_d)$, meaning for each word j in document d, we sample a single topic $z_{d,j}$ from the categorical distribution parameterized by θ_d. With these assumptions in mind, we can write out the conditional probability of the set of all topic assignments given the set of document-topic distributions:

$$P(Z|\Theta) = \prod_{d=1}^{D} \left(\prod_{j=1}^{N_d} p(z_{d,j}|\theta_d) \right) \tag{3}$$

Since $z_{d,j}|\theta_d \sim Cat(\theta_d)$, we have that $p(z_{d,j}) = \prod_{k=1}^{K} (\theta_{d,k})^{x_k{}^j}$ where for a given word, j, out of $\{x_{k=1}^j, \cdots, x_{k=k'}^j, \cdots, x_{k=K}^j\}$, exactly one of $x_{k=k'}^j = 1$ and the rest are all zero, i.e., $x_k^j = 0$ where $k \neq k'$, so word j belongs to just one topic. This stems from the categorical distribution because we are sampling a single topic for the j^{th} word in document d. The sampled topic k in document d has probability $\theta_{d,k}$.

Thus,

$$\prod_{j=1}^{N_d} p(z_{d,j}|\theta_d) = \prod_{j=1}^{N_d} \left(\prod_{k=1}^{K} (\theta_{d,k})^{x_k^j} \right) = \prod_{k=1}^{K} (\theta_{d,k})^{\sum_{j=1}^{N_d} x_k^j} \tag{4}$$

Now, $\sum_{j=1}^{N_d} x_k^j$ represents the number of words in document d that were assigned to topic k. We will denote this count as $C(d,k)$ or C_d^k, i.e.,

$$C(d,k) = C_d^k = \sum_{j=1}^{N_d} x_k^j$$

We can rewrite equation 4 using this new notation:

$$\prod_{j=1}^{N_d} p(z_{d,j}|\theta_d) = \prod_{k=1}^{K} (\theta_{d,k})^{\sum_{j=1}^{N_d} x_k^j} = \prod_{k=1}^{K} (\theta_{d,k})^{C(d,k)} \tag{5}$$

Using the new notation in equation 5, we can substitute into equation 3 to rewrite the equation for the conditional probability of the set of topic assignments given the set of document-topic distributions:

$$P(Z|\Theta) = \prod_{d=1}^{D} \left(\prod_{j=1}^{N_d} p(z_{d,j}|\theta_d) \right) = \prod_{d=1}^{D} \left(\prod_{k=1}^{K} (\theta_{d,k})^{C(d,k)} \right) \tag{6}$$

We can now use equation 6 and equation 2 to calculate the probability of the set of topic assignments:

$$I_2 = P(Z) = \int P(Z|\Theta)P(\Theta)\, d\Theta$$

$$P(Z) = \int \left[\prod_{d=1}^{D}\left(\prod_{k=1}^{K}(\theta_{d,k})^{C(d,k)}\right)\right]\left[\prod_{d=1}^{D}\left(\frac{1}{B(\boldsymbol{\alpha})}\prod_{k=1}^{K}\left((\theta_{d,k})^{\alpha-1}\right)\right)\right] d\Theta \quad (7)$$

Since the document-topic distributions, θ_d, are independent of each other, the integral over Θ can be decomposed into separate integrals for each document. Thus, we can rewrite this equation as a product over all documents in the corpus.

$$P(Z) = \frac{1}{B(\boldsymbol{\alpha})}\prod_{d=1}^{D}\left[\int\left(\prod_{k=1}^{K}\left((\theta_{d,k})^{\alpha-1+C(d,k)}\right)\right)d\theta_d\right] \quad (8)$$

Using equation 1, we can manipulate this equation to set the integral equal to the B function with parameter $\boldsymbol{\alpha} + \boldsymbol{C_d}$.

$$\int\left(\prod_{k=1}^{K}\left((\theta_{d,k})^{\alpha-1+C(d,k)}\right)\right)d\theta_d = B(\boldsymbol{\alpha} + \boldsymbol{C_d})$$

where

$$\boldsymbol{C_d} = \{C_d^{k=1}, C_d^{k=2}, \cdots, C_d^{k=K}\}$$

and

$$B(\boldsymbol{\alpha} + \boldsymbol{C_d}) = \frac{\prod_{k=1}^{K}\Gamma(\alpha_k + C_d^k)}{\Gamma(\sum_{k=1}^{K}(\alpha_k + C_d^k))}$$

Simplifying the probability of the topic assignments in equation 8 using B notation, we get:

$$I_2 = P(Z) = \int P(Z|\Theta)P(\Theta)\,d\Theta = \frac{1}{B(\boldsymbol{\alpha})}\prod_{d=1}^{D} B(\boldsymbol{\alpha} + \boldsymbol{C_d}) \tag{9}$$

Now we will derive I_1, the conditional probability of the observed words W given the latent topic assignments Z. We follow a similar process used to derive I_2.

Recall

$$I_1 = P(W|Z) = \int P(W|Z,\Phi)P(\Phi)\,d\Phi$$

Note that $\Phi = \{\phi_k\} = \{\phi_1, \cdots, \phi_K\}$ is a K x V matrix where K represents the set of topics, and V represents the set of all words. Observe that each row $\phi_k = \{\phi_{k,1}, \phi_{k,2}, \cdots, \phi_{k,V}\}$ is the word distribution for topic k, and each element $\phi_{k,v}$ of the matrix Φ represents the probability of choosing word v given topic k. For example, $\phi_{k,1}$ represents the probability of choosing word 1 given topic k.

Let's assume $\phi_k \sim \text{Dirichlet}(\boldsymbol{\beta})$, meaning that for each topic k, we draw its word distribution ϕ_k from a Dirichlet distribution with some parameter vector $\boldsymbol{\beta}$. We also assume that the topic-word distributions are independent across topics. We can now write the probability of the set of topic-word distributions as the product of probabilities representing each individual topic-word distribution given the Dirichlet parameter vector $\boldsymbol{\beta}$:

$$P(\Phi) = \prod_{k=1}^{K} P(\phi_k|\boldsymbol{\beta})$$

Using the PDF of the Dirichlet distribution with parameter vector $\boldsymbol{\beta}$ we can expand the equation for the probability of topic-word distributions:

$$P(\Phi) = \prod_{k=1}^{K}\left(\frac{1}{B(\boldsymbol{\beta})}\prod_{v=1}^{V}\left((\phi_{k,v})^{\beta-1}\right)\right) \tag{10}$$

We assume that each word $w_{d,j}$ is generated only based on its assigned topic and does not depend on the other words in document d. Further, we assume $w_{d,j}|\phi_k \sim Cat(\phi_k)$, meaning given that a word belongs to topic k, its probability is determined by the categorical distribution parameterized by ϕ_k. In general, each word in a document is assumed to be generated by first selecting a topic and then sampling a word from that topic's word distribution ϕ_k. Because one word is assigned to the topic, this follows the categorical distribution.

Given these assumptions, we can write the conditional probability of the observed words given their topic assignments and topic-word distribution:

$$P(W|Z,\Phi) = \prod_{d=1}^{D}\left(\prod_{j=1}^{N_d}\left(\prod_{k=1}^{K} p(w_{d,j}|\phi_k)\right)\right) \tag{11}$$

Since $w_{d,j}|\phi_k \sim Cat(\phi_k)$, we have that $p(w_{d,j}|\phi_k) = \prod_{v=1}^{V}(\phi_{k,v})^{x_v^{d,j}}$ where for a given word, j, in document d, out of $\{x_{v=1}^{d,j}, \cdots, x_{v=v'}^{d,j}, \cdots, x_{v=V}^{d,j}\}$, exactly one of $x_{v=v'}^{d,j} = 1$ and the rest are all zero, i.e., $x_v^{d,j} = 0$ where $v \neq v'$, as we are drawing a word v from the topic-word distribution ϕ_k with probability $\phi_{k,v}$.

Continuing from equation 11 and substituting $p(w_{d,j}|\phi_k) = \prod_{v=1}^{V}(\phi_{k,v})^{x_v^{d,j}}$, we get

$$P(W|Z,\Phi) = \prod_{d=1}^{D}\left(\prod_{j=1}^{N_d}\left(\prod_{k=1}^{K}\prod_{v=1}^{V}(\phi_{k,v})^{x_v^{d,j}}\right)\right) = \prod_{k=1}^{K}\prod_{v=1}^{V}(\phi_{k,v})^{\sum_d \sum_j x_v^{d,j}}$$

Now, $\sum_d \sum_j x_v^{d,j}$ represents the number of times word v was assigned to topic k. We will denote this count as C_k^v.

$$C(k,v) = C_k^v = \sum_d \sum_j x_v^{d,j}$$

Using this new notation, we can rewrite equation 11 as:

$$P(W|Z,\Phi) = \prod_{k=1}^{K}\prod_{v=1}^{V}(\phi_{k,v}^{C(k,v)}) \tag{12}$$

To find I_1, the conditional probability of the observed words W given the topic assignments Z, we can plug in the probability of the set of topic-word distributions in equation 10 as well as equation 12 into the integral:

$$I_1 = P(W|Z) = \int P(W|Z,\Phi)P(\Phi)\,d\Phi$$

$$P(W|Z) = \int \left(\prod_{k=1}^{K}\prod_{v=1}^{V}(\phi_{k,v}^{C(k,v)})\right)\prod_{k=1}^{K}\left(\frac{1}{B(\boldsymbol{\beta})}\prod_{v=1}^{V}((\phi_{k,v})^{\beta-1})\right)$$

Noting that the topic-word distributions ϕ_k are independent of each other, the integral over Φ can be decomposed into separate integrals for each topic.

Thus, we rewrite this equation as a product of integrals over all topics.

$$P(W|Z) = \frac{1}{B(\boldsymbol{\beta})}\prod_{k=1}^{K}\left(\int\prod_{v=1}^{V}(\phi_{k,v})^{\beta+C(k,v)-1}d\phi_k\right)$$

Using equation 1, we can set the integral equal to the B function parameterized by $\boldsymbol{\beta} + \boldsymbol{C_k}$.

$$\int\prod_{v=1}^{V}(\phi_{k,v})^{\beta+C(k,v)-1}d\phi_k = B(\boldsymbol{\beta} + \boldsymbol{C_k})$$

where

$$\boldsymbol{C_k} = \{C_k^{v=1}, C_k^{v=2}, \cdots, C_k^{v=V}\}$$

Thus, the conditional probability of the observed words W given the topic assignments Z can be rewritten using the B notation:

$$I_1 = P(W|Z) = \int P(W|Z,\Phi)P(\Phi)\,d\Phi = \frac{1}{B(\boldsymbol{\beta})}\prod_{k=1}^{K} B(\boldsymbol{\beta} + \boldsymbol{C_k}) \tag{13}$$

Using the probability of topic assignments in equation 9 and the conditional probability of the observed words given the topic assignments in equation 13, we can now write the full joint distribution of the observed words and topic assignments as:

$$\begin{aligned} P(W,Z;\alpha,\beta) &= P(W|Z)P(Z) = I_1 \times I_2 \\ &= \left[\frac{1}{B(\boldsymbol{\alpha})}\prod_{d=1}^{D} B(\boldsymbol{\alpha} + \boldsymbol{C_d})\right]\left[\frac{1}{B(\boldsymbol{\beta})}\prod_{k=1}^{K} B(\boldsymbol{\beta} + \boldsymbol{C_k})\right] \end{aligned} \tag{14}$$

3.2.2 Gibbs Sampler

Now that we have derived the joint distribution of words and topics, we have all we need to derive the conditional distribution of an individual topic for the Gibbs sampler.

Let $Z = \{z_{d,n}\} = \{z_{d=1,j=1}, \cdots, z_{d=1,j=N_1}, \cdots, z_{d=D,j=N_D}\}$ be a $\sum_{d=1}^{D} N_d$ dimensional vector where Z denotes the collection of all latent topic variables $z_{d,n}$ corresponding to all words in all documents.

Let's also assume we have a Markov Chain $X =< Z^{(0)}, Z^{(1)}, \cdots, Z^{(N_{iter})} >$ over the data whose stationary distribution converges to the posterior on the distribution of Z. Further, let $Z = \{z_{d,n}\} = \{z_i\}$ for ease of notation.

For a given token, $i = (d', j')$, representing the word j' in document d', the Gibbs sampler estimates the conditional probability that the latent topic variable z_i at token $i = (d', j')$ is assigned to topic $k' \in \{1, \cdots, K\}$ having observed all other topic assignments and words. Observe that the conditional

probability used for Gibbs sampling is simply a fraction of joint probabilities, one of which we have already derived (the numerator).

$$P(z_i = k'|Z_{-i}, W) = \frac{P(Z, W)}{P(Z_{-i}, W)} = \frac{P(W|Z)P(Z)}{P(W_{-i}|Z_{-i})P(Z_{-i})p(w_i)}$$

Note that $Z = \{z_i, Z_{-i}\}$ and $W = \{w_i, W_{-i}\}$. To understand the second equivalence, specifically the denominator, we can first break down:

$$P(Z_{-i}, W) = P(W|Z_{-i})P(Z_{-i})$$

using Bayes Rule.

Then, using the definition of conditional independence, which is when two random variables are independent of each other given the presence of a third variable, we have:

$$P(X, Y|Z) = P(X|Z)P(Y|Z)$$

so

$$P(W|Z_{-i}) = P(W_{-i}, w_i|Z_{-i}) = P(W_{-i}|Z_{-i})P(w_i|Z_{-i}) = P(W_{-i}|Z_{-i})P(w_i)$$

where w_{-i} and w_i are independent of each other given Z_{-i} in the first equality, and w_i is independent of Z_i in the second equality because the word at token i is independent of the topic assignments at all other tokens. This follows from a previously referenced assumption that each word is generated only based on its assigned topic and does not depend on other words.

Thus,

$$P(W|Z_{-i}) = P(W_{-i}|Z_{-i})p(w_i)$$

so

$$P(Z_{-i}, W) = P(W_{-i}|Z_{-i})P(Z_{-i})p(w_i)$$

Now expanding the Gibbs sampling conditional probability,

$$P(z_i = k'|Z_{-i}, W) = \frac{P(W|Z)P(Z)}{P(W_{-i}|Z_{-i})P(Z_{-i})p(w_i)} \propto \frac{P(W|Z)P(Z)}{P(W_{-i}|Z_{-i})P(Z_{-i})} \quad (15)$$

Note that the subscript $-i$ denotes all values upon discounting the token at w_i.

Expanding the Gibbs sampler conditional probability in equation 15 using the joint distribution derivation in equation 14,

$$P(z_i = k'|Z_{-i}, W_{-i}, w_i) \propto \frac{P(W|Z)P(Z)}{P(W_{-i}|Z_{-i})P(Z_{-i})} = \frac{[\prod_{d=1}^{D} B(\boldsymbol{\alpha} + \boldsymbol{C_d})][\prod_{k=1}^{K} B(\boldsymbol{\beta} + \boldsymbol{C_k})]}{[\prod_{d=1}^{D} B(\boldsymbol{\alpha} + \boldsymbol{C_d})][\prod_{k=1}^{K} B(\boldsymbol{\beta} + \boldsymbol{C_k})]_{-i}}$$

$$= \left[\frac{\prod_{d=1}^{D} B(\boldsymbol{\alpha} + \boldsymbol{C_d})}{\prod_{d=1}^{D} B(\boldsymbol{\alpha} + \boldsymbol{C_d})_{-i}}\right] \times \left[\frac{\prod_{k=1}^{K} B(\boldsymbol{\beta} + \boldsymbol{C_k})}{\prod_{k=1}^{K} B(\boldsymbol{\beta} + \boldsymbol{C_k})_{-i}}\right] \quad (16)$$

Thus, the conditional distribution that the i^{th} token will have topic k' is proportional to the full joint distribution of the model divided by the joint distribution considering that the token at w_i and its corresponding topic assignment did not in exist in the data.

Observing that $\boldsymbol{\alpha}$ remains fixed and i corresponds to the topic and words $\{z_i, w_i = (d', j')\}$ at some document d' and some position j' in that document, we can simplify the first term in (15) by canceling the product across all documents and only considering the document d':

$$\left[\frac{\prod_{d=1}^{D} B(\boldsymbol{\alpha} + \boldsymbol{C_d})}{\prod_{d=1}^{D} B(\boldsymbol{\alpha} + \boldsymbol{C_d})_{-i}}\right] = \frac{B(\boldsymbol{\alpha} + \boldsymbol{C_{d=d'}})}{B(\boldsymbol{\alpha} + \boldsymbol{C_{d=d'}})_{-i}} \quad (17)$$

We are able to simplify because for all documents other than d', the numerator and the denominator cancel out, as w_i will not be excluded in the denominator.

Now, we observe what happens to $C_{d=d'}$ with or without including the term at $w_i = (d', j')$ whose corresponding topic assignment is $z_i = k'$. Recall that $\boldsymbol{C_d} = \{C_d^{k=1}, C_d^{k=2}, \cdots, C_d^{k=K}\}$ and $C_d^k = C(d,k)$ is the number of words in document d that are assigned to topic k. We observe:

$$C(d,k) = \begin{cases} [C(d,k)]_{-i}, & \text{if } k \neq k', \\ [C(d,k)]_{-i} + 1, & \text{if } k = k'. \end{cases} \tag{18}$$

because the number of words in document d that are assigned to topic k increases by one when we consider the i^{th} token, where $k = k'$.

Expanding equation 17 by plugging in the definition of $B(\boldsymbol{\alpha}) = \frac{\prod \Gamma(\alpha_i)}{\Gamma(\sum \alpha_i)}$, we get:

$$\frac{B(\boldsymbol{\alpha} + \boldsymbol{C_{d=d'}})}{B(\boldsymbol{\alpha} + [\boldsymbol{C_{d=d'}}]_{-i})} = \frac{\prod_{k=1}^{K} \Gamma\big(\alpha + C(d',k)\big)}{\prod_{k=1}^{K} \Gamma\big(\alpha + C(d',k)_{-i}\big)} \times \frac{\Gamma\Big(\sum_{k=1}^{K}\big(\alpha + C(d',k)_{-i}\big)\Big)}{\Gamma\Big(\sum_{k=1}^{K}\big(\alpha + C(d',k)\big)\Big)}$$

Using the delineation noted in equation 18, this equation further simplifies to (upon canceling out all non-k' terms):

$$= \frac{\prod_{k=1}^{K} \Gamma\big(\alpha + C(d',k)\big)}{\prod_{k=1}^{K} \Gamma\big(\alpha + C(d',k)_{-i}\big)} \times \frac{\Gamma\left(\sum_{k=1\setminus k'}^{K}\big(\alpha + C(d',k)_{-i}\big) + [\big(\alpha + C(d',k')_{-i}\big)]\right)}{\Gamma\left(\sum_{k=1\setminus k'}^{K}\big(\alpha + C(d',k)\big) + [\big(\alpha + C(d',k')\big)]\right)}.$$

$$= \frac{\Gamma\big(\alpha + C(d',k) + 1\big)}{\Gamma\big(\alpha + C(d',k)_{-i}\big)} \times \frac{\Gamma\left(\sum_{k=1\setminus k'}^{K}\big(\alpha + C(d',k)_{-i}\big) + [\big(\alpha + C(d',k')_{-i}\big)]\right)}{\Gamma\left(\sum_{k=1\setminus k'}^{K}\big(\alpha + C(d',k)\big) + [\big(\alpha + C(d',k')\big) + 1]\right)}.$$

Using the identity $\Gamma(x+1) = x\Gamma(x)$, we get:

$$\frac{B(\boldsymbol{\alpha} + \boldsymbol{C_{d=d'}})}{B(\boldsymbol{\alpha} + [\boldsymbol{C_{d=d'}}]_{-i})} = \frac{\alpha + C(d',k')_{-i}}{\sum_{k=1}^{K}\big(\alpha + C(d',k)_{-i}\big)}$$

Similarly, we can simplify the second term in equation 16:

$$\frac{\prod_{k=1}^{K} B(\boldsymbol{\beta} + \boldsymbol{C_k})}{\prod_{k=1}^{K} B(\boldsymbol{\beta} + \boldsymbol{C_k})_{-i}} = \frac{B(\boldsymbol{\beta} + \boldsymbol{C_{k=k'}})}{B(\boldsymbol{\beta} + \boldsymbol{C_{k=k'}})_{-i}} = \frac{\beta + C(k', v')_{-i}}{\sum_{v=1}^{V} \big(\beta + C(k', v)_{-i}\big)}$$

where v' refers to the token, which is assigned to w_i.

We can thus simplify the Gibbs sampling update equation for LDA as follows:

$$P(z_i = k' \mid Z_{-i}, W_{-i}, w_i) \propto$$

$$\left[\frac{\alpha + C(d', k')_{-i}}{\sum_{k=1}^{K}(\alpha + C(d', k)_{-i})}\right] \cdot \left[\frac{\beta + C(k', v')_{-i}}{\sum_{v=1}^{V}(\beta + C(k', v)_{-i})}\right] \tag{19}$$

Sampling from this conditional distribution repeatedly is how Gibbs sampling approximates the posterior distribution for LDA. Note that this conditional probability is not the LDA model's full joint distribution, nor the general Gibbs sampling algorithm for all contexts, but specifically the Gibbs sampling method for LDA.

3.3 Posterior on θ and ϕ

We can next compute the posterior distributions for θ_d and ϕ_k. These posterior distributions are useful because they allow us to make informed estimates about the distribution of topics in documents as well as how topics can generate words.

First, we compute the posterior distribution for θ_d having observed topic assignments $z_{d,n}$ in document d.

We know that the prior $\theta_d|\boldsymbol{\alpha} \sim \text{Dirichlet}(\boldsymbol{\alpha})$, so the prior probability looks like:

$$P(\theta_d) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{K} \left((\theta_{d,k})^{\alpha - 1} \right)$$

We also know that the likelihood is distributed as a categorical distribution $z_{d,j}|\theta_d \sim Cat(\theta_d)$, so the likelihood of document d given θ_d is:

$$P(Z_d|\theta_d) = \prod_{j=1}^{N_d} P(z_{d,j}|\theta_d) = \prod_{k=1}^{K} (\theta_{d,k})^{C(d,k)}$$

using equation 3 and equation 5 to calculate the respective equalities.

Thus, using Bayes Theorem, the posterior on θ_d is:

$$P(\theta_d|Z_d) = \frac{P(Z_d|\theta_d)P(\theta_d)}{\int (P(Z_d|\theta_d)P(\theta_d))d\theta_d} \propto \prod_{k=1}^{K} (\theta_{d,k})^{C(d,k)+\alpha-1}$$

revealing that the posterior on θ_d follows a Dirichlet distribution parameterized by $\boldsymbol{\alpha} + \boldsymbol{C_d}$, where $\theta_d|Z_d \sim Dir(\boldsymbol{\alpha} + \boldsymbol{C_d})$

We observe that both the prior of the θ_d random variable and the posterior are proportional to a Dirichlet distribution, demonstrating that the prior is, in fact, a conjugate prior to the categorical distribution.

We can now find the expected value of the Dirichlet distributions. These expectations are extremely useful because they can be used to estimate the final posterior estimates of the document-topic distribution θ_d and the topic-word distribution ϕ_k after Gibbs sampling stabilizes.

Recall that the expected value of a Dirichlet distribution is given by the following:

$$E[X_i] = \frac{\alpha_i}{\alpha_0}$$

where $\alpha_0 = \sum_{k=1}^{K} \alpha_i$

Thus, the expected value of the probability mass associated to each topic in document d is:

$$E[\theta_{d,k}] = \frac{\alpha + C(d,k)}{\sum_{k=1}^{K}(\alpha + C(d,k))} \tag{20}$$

Equation 20 represents the posterior mean of the proportion of topic k in document d. After inference (Gibbs sampling), we can interpret this value as a point-estimate representation of the mixture of topics in each document, which is crucial for topic-modeling applications such as document classification and similarity search (used later in Chapter 5).

Following a similar process used to derive equation 20, we can compute the posterior distribution for ϕ_k. We observe that $\phi_k | W_k \sim Dir(\boldsymbol{\beta} + \boldsymbol{C_k})$, so:

$$E[\phi_{k,v}] = \frac{\beta + C(k,v)}{\sum_{v=1}^{V}(\beta + C(k,v))} \tag{21}$$

Equation 21 represents the posterior mean of the probability that topic k generates word v. This value provides a concrete topic–word distribution for each topic, which is useful for interpreting topics.

Chapter 4

Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) is a framework that enhances the output of a large language model by combining an information retrieval system with a generative language model. Unlike other language models, which rely solely on the pre-trained model (parametric memory) to generate outputs, RAG also leverages an external knowledge base (non-parametric memory) to create a "hybrid" model that improves the quality of outputs. An example of an external knowledge base is a collection of Wikipedia articles such as the one used in Lewis et al. (2020), which differs from the data used in the pre-trained model.

There are several ways in which RAG can improve the quality of LLM outputs. First, RAG enables the model to learn up-to-date information. For prompts that require knowledge of the latest news, the original data used to train the model (parametric memory) would be irrelevant, as the data is outdated. However, incorporating non-parametric memory via RAG enables the model to maintain relevancy by incorporating data from dynamically-updated sources. Second, RAG promotes scalable domain adaptation by simply swapping the external knowledge base, which can be particularly useful for generating responses that serve specialized domains such as the legal or medical field. Lastly, and perhaps most importantly, by explicitly conditioning the model to generate responses based on real documents, RAG can reduce model "hallucinations," which refers to responses in which the model generates incorrect or unverifiable information. Overall, RAG has the

potential to drastically improve language models by dynamically updating them to reflect changes in the world, making it easier to scale them to reflect domain-specific knowledge, and reducing hallucinations.

RAG can be thought of as a hybrid approach with two parts: a retriever (which includes the non-parametric memory) and a generator (which includes the parametric memory). Below is a visualization from Lewis et al. (2020) that shows the RAG pipeline:

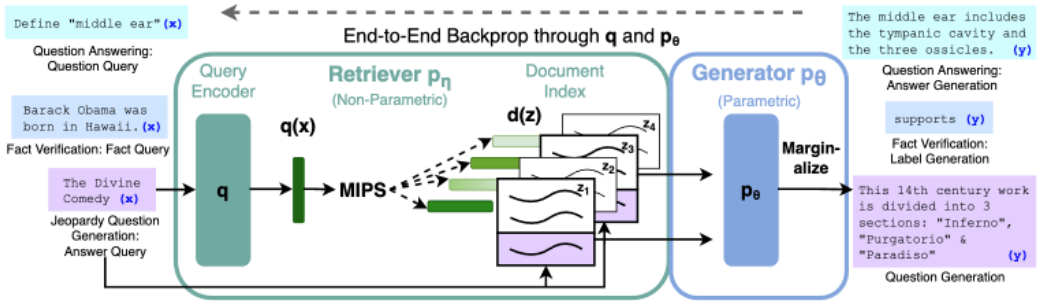

Figure 4.1: Overview of RAG-based approach. RAG incorporates a retrieval system p_η and a generator p_θ to generate a response y to a some query x.

On the left is the retrieval system p_η, which leverages dense passage retrieval (DPR). First, a query x is passed into a query encoder q, which transforms the query into a query embedding vector $\mathbf{q}(x)$. Then, each of the documents z_i in the corpus are passed through a document encoder (not shown) and transformed into document embedding vectors $\mathbf{d}(z)$. Then, Maximum Inner Product Search (MIPS) is performed among all documents to find the top-k documents to retrieve z by maximizing the inner product between $\mathbf{q}(x)$ and $\mathbf{d}(z)$. The retrieval system is represented as follows:

$$p_\eta(z|x) \propto \exp(\mathbf{d}(z)^\top \mathbf{q}(x))$$

On the right is the generator component $p_\theta(y_i|x, z, y_{1:i-1})$, which takes the query x, retrieved documents z, and the previous token $y_{1:i-1}$ as inputs and generates some response y_i.

In my thesis, I emulate the model approach in Lewis et al. (2020), in which the authors leverage both pre-trained parametric and non-parametric memory. In this paper, the authors incorporate both parametric and non-parametric memory components that were pre-trained and pre-loaded with extensive knowledge. In particular, they created a model where the parametric memory is a pre-trained sequence-to-sequence (seq2seq) transformer model called BART, and the non-parametric memory is a dense vector index of 21 million Wikipedia articles. Although my RAG framework uses different models than those in the paper, the overall process is similar (described further in Chapter 5).

4.1 Dense Passage Retrieval

The non-parametric memory in the RAG framework is accessed with a retriever called Dense Passage Retriever (DPR), which incorporates a dense-retrieval system. Introduced in Karpukhin et al. (2020), DPR is able to learn semantic relationships between words, grouping synonyms and paraphrases. This differs from term-based systems like tf-idf or BM25, which simply match queries and passages based on specific words and are unable to capture underlying semantic relationships. Mathematically, a retriever can be represented as a function that takes a question x and a corpus C as inputs and returns a much smaller filter set of texts $C_F \subset C$ as the output, where $|C_F| = k \ll |C|$

$$R : (x, C) \rightarrow C_F$$

This hyperparameter k is tuned during the retrieval process to optimize the number of documents to retrieve when generating a response.

The goal of DPR is to index all text documents to a low-dimensional and continuous space such that it can efficiently retrieve the top k most relevant passages to the input query from the entire corpus of M passages, providing useful information that aids the model in generating a response. DPR uses two separate encoders in the retrieval process, a passage encoder $\mathbf{d}(\cdot)$ and a question encoder $\mathbf{q}(\cdot)$. The passage encoder $\mathbf{d}(\cdot)$ maps any chunk of text to a d-dimensional vector and creates an index for all M passages used for retrieval. The question encoder $\mathbf{q}(\cdot)$ maps an input question to a d-dimensional vector as well but retrieves the k most relevant passages, that is, the passages

which share the most similar vector representations to the question vector. Similarity is defined using the dot product of vectors:

$$sim(x, z) = \mathbf{q}(x)^\top \mathbf{d}(z)$$

In the next sections, I explore a potential improvement to the retrieval mechanism in RAG. I hypothesize that by incorporating LDA as a pre-processing "filtering" step before retrieval, I can reduce the search space of documents considered for retrieval. Mathematically, this new retrieval system can be represented as:

$$R_{\text{LDA}} : (x, C_{\text{LDA}}) \to C_f$$

where C_{LDA} is the remaining corpus of documents after LDA filtering has been conducted and C_f is the final filter set of the top-k documents, where $|C_f| = k$. Observe that $C_{\text{LDA}} \subset C$ and $|C_{\text{LDA}}| \leq |C|$. The extent to which $|C_{\text{LDA}}|$ is less than $|C|$ depends on the topic-similarity threshold, which is described in Chapter 5.

Further, note that even though the number of retrieved documents from the new filter set is the same as before, $|C_f| = |C_F| = k$, the exact documents chosen for retrieval likely differ. This is the basis of my hypothesis. I conjecture that by reducing the search space from C to C_{LDA}, I can find a filter set of documents C_f that has the potential to improve both model efficiency and response quality.

Chapter 5

Methods

This experiment compares the question answering performance of an RAG-only model against an RAG model that incorporates LDA as a pre-processing step. Question answering (QA) tasks have served as one of the most popular benchmarks for language model evaluation, enabling researchers to evaluate an LLM's ability to locate, comprehend, and reason over text. There are two main approaches commonly considered in question answering: closed-book QA and open-domain QA. In a closed-book QA model, the model only sees the question and the provided context, which contains the answer. In an open-domain QA model, the model considers a large corpus of text, which contains the correct answer. In this experiment, I explore whether I can extend the capabilities of the *deepset/roberta-base-squad2* model from that of a closed-book QA model to an open-domain QA model.

5.1 Choosing Models

For this experiment, I use both the dual-encoder model *all-MiniLM-L6-v2* Wang et al. (2020) and the encoder-only model *roberta-base-squad2* Liu et al. (2019) to comprise the parametric memory for an RAG-based model. In particular, I use *all-MiniLM-L6-v2* to transform each passage of text into a 384-dimensional embedding vector. Designed as a sentence and short paragraph encoder, this model converts short passages into embedding vectors that capture semantic relationships in the text. In addition, I use the *roberta-base-squad2* model to generate responses by extracting answer spans from

the text. Fine-tuned on *SQuAD 2.0* Rajpurkar et al. (2018), a reading comprehension dataset consisting of questions posed by crowdworkers on a set of Wikipedia articles, this model performs well on extractive QA tasks.

For the non-parametric memory, I incorporate the *databricks-dolly-15k* dataset Conover et al. (2023), an open source dataset of instruction-following records generated by Databricks employees in several categories such as summarization, QA, and information extraction. Additionally, I use Facebook AI Similarity Search (FAISS) Douze et al. (2024) to efficiently compute an inner product search for the most relevant documents.

5.2 LDA Filtering Step

To incorporate the LDA pre-processing step, I introduce a topic-similarity threshold, which computes the cosine similarity between topic-distribution vectors and only keeps those whose score meets the threshold, serving as a filtering step. Cosine similarity measures how similar two non-zero vectors are in a multi-dimensional space. In this context, cosine similarity measures how similar the topic-distribution of each document in the corpus is to the topic-distribution of the question. Cosine similarity is calculated as follows:

$$sim(\mathbf{q}, \mathbf{d}^{(i)}) = \frac{\mathbf{q} \cdot \mathbf{d}^{(i)}}{||\mathbf{q}||\,||\mathbf{d}^{(i)}||}$$

where $\mathbf{q} = (q_1, q_2, \cdots, q_k, \cdots, q_K)$ is the question's topic-distribution vector and $\mathbf{d}^{(i)} = (d_1^{(i)}, d_2^{(i)}, \cdots, d_k^{(i)}, \cdots, d_K^{(i)})$ is the i^{th} document's topic-distribution vector. Note that q_k represents the probability that topic k is present in the question, and $d_k^{(i)}$ represents the probability that topic k is in document i. We also assume that the number of topics equals K (I experiment with various values when tuning this hyperparameter, which is described later in this section). A high cosine similarity value would indicate that the question and the document have similar topic distributions, whereas a low value would indicate that the question and document are dissimilar.

5.3 F_1 Score

To evaluate model responses, I use a metric called F_1 score, which measures the harmonic mean of precision and recall. Precision measures the fraction of values that belong to a positive class (true positives) out of all the values which are predicted to belong to the positive class (true positives + false positives). In the context of this experiment, precision essentially measures how many tokens belong to the correct answer out of all the tokens generated in the model's response. Precision is calculated below:

$$\text{Precision} = \frac{\text{Number of True Positives (TP)}}{\text{Number of True Positives (TP)} + \text{Number of False Positives (FP)}}$$

Recall measures the fraction of values predicted to be positive (true positives) out of all values that truly belong to the positive class (true positives + false negatives). In this context, recall measures how many tokens were successfully predicted out of all the tokens in the correct answer. Recall is calculated below:

$$\text{Recall} = \frac{\text{Number of True Positives (TP)}}{\text{Number of True Positives (TP)} + \text{Number of False Negatives (FN)}}$$

F_1 score incorporates both precision and recall and calculates their harmonic mean:

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

F_1 score provides a metric that measures a model's ability to generate correct tokens, essentially matching the tokens in the correct answer (recall), while avoiding spurious ones (precision), making it a useful metric for model evaluation. By not fully penalizing incorrect tokens, this metric is both more flexible and more informative than more rigid metrics like exact match.

5.4 Tuning Model Hyperparameters

In creating the RAG model that incorporated LDA (which I refer to as LDA+RAG), there were three primary hyperparameters to tune: the number of topics (*num_topics*), the topic-similarity threshold (*threshold*), and the number of documents to retrieve (k). To choose the best hyperparameters for the LDA+RAG model, I performed the following grid search and then selected the configuration that resulted in the highest F_1 score:

- *num_topics*: {10, 20, 30}
- *threshold*: {0.3, 0.5, 0.7}
- k: {10, 50, 100}

I intentionally kept the range for the number of topics relatively small to encourage interpetability; the more topics, the less interpretable each topic is. Further, I experimented with various threshold values and decided to limit the highest threshold to 0.7 because when I experimented with a higher threshold value of 0.9, there were some queries for which no documents passed the threshold. Lastly, I chose the range of k values based on Karpukhin et al. (2020), which observed that the optimal value of k was usually small, where $k \leq 100$. I then compared results from the LDA+RAG model to an RAG-only baseline model that uses the same respective k value. For example, if the grid search for the LDA+RAG model found that $k = 10$ was the optimal top-k value for a certain sample of documents, I would use this same value for the RAG-baseline model, allowing for fair comparisons.

5.5 High-level Pipeline

Below is a high-level pipeline of the process used to create and compare the RAG-based models.

1. **Document Preparation**
    - Split text corpus into manageable chunks (documents)
    - Embed all documents into a standard vector index (FAISS) for a "baseline" retriever

2. **Topic Modeling for Filtering**

   - Train an LDA model on these documents to get a topic distribution for each
   - Infer the query's own topic-distribution vector via the same LDA

3. **Enhanced Retrieval Pipeline**

   - *LDA filter*: Compute cosine similarity between the query's topic-distribution vector and each document's topic-distribution vector; discard anything below a chosen threshold
   - *FAISS*: Build a temporary FAISS index over only the remaining topically relevant documents C_{LDA} and retrieve the top-k documents C_f from that smaller set

$$R_{\text{LDA}} : (x, C_{\text{LDA}}) \to C_f$$

4. **Answer Generation**

   - Concatenate query with the k most relevant documents
   - Pass this into the QA model to generate an answer

5. **Comparison to Baseline RAG Model**

   - Run the same QA model over the FAISS retriever for the entire corpus without filtering C and retrieve the top-k documents C_F (baseline RAG-only model)

$$R : (x, C) \to C_F$$

   - Compare F_1 score and runtime between LDA+RAG model vs baseline RAG model

Chapter 6

Results

6.1 Results on Abridged SQuAD 2.0 Dataset

First, I evaluate the LDA+RAG model on an abridged version of the *SQuAD 2.0* dataset to serve as a proof of concept. Because the entire version of *SQuAD 2.0* includes entries in which the correct answer is an empty string, it is considerably more difficult for the model to perform well because the model always outputs some response, automatically penalizing it for all such entries. Thus, I reduced the dataset to only include entries where the answer is a non-empty string of text.

Because the abridged *SQuAD 2.0* dataset is still quite large (11.9k rows), I chose to evaluate the model on random samples from *SQuAD 2.0*, each with size $n = 100$. To ensure these samples are reproducible, I chose three random seeds. I then performed a grid search over the model's hyperparameters (as described in Chapter 5) and selected the configuration that produced the highest F_1 score for each seed. The optimal configurations for each sample are shown below, where s represents each seed:

Table 6.1: Best hyperparameter configurations for LDA+RAG model

	num_topics	*threshold*	k	F_1 score
$s = 1$	20	0.7	10	84.64
$s = 47$	20	0.5	50	87.39
$s = 99$	30	0.7	100	83.19

We can observe some patterns from these configurations. First, it seems as if the optimal number of topics chosen varies between 20 to 30, suggesting that 10 topics might be too few. This could be due to the size of the *databricks-dolly-15k* dataset, which covers a wide variety of subjects, leading to better performance over more topics. Second, the optimal value for the topic-similarity threshold appears to be between 0.5 and 0.7, indicating that a relatively high threshold leads to better performance. This result supports our hypothesis that retrieving topics based on higher levels of topic similarity contributes to a higher F_1 score, which we will expand upon later. Lastly, the optimal values of k differ for each sample, suggesting that k should be tuned on a case-by-case basis.

I next compare the average F_1 score of the LDA+RAG model against the RAG-only baseline model (using the same k values) to determine the effect of the LDA pre-processing step on performance. I exclude results from the *roberta-base-squad2* model in this section. The results are shown below:

Table 6.2: F_1 score by model

	LDA+RAG	RAG
$s = 1$	**84.64**	81.19
$s = 47$	**87.39**	78.49
$s = 99$	**83.19**	69.52

We observe that the LDA+RAG model outperforms the RAG baseline model in every sample, demonstrating the model's robustness across varying sets of questions. These results suggest that incorporating LDA as a pre-processing step leads to a higher F_1 score on average, indicating that incorporating LDA improves model performance. By incorporating LDA as a filter before retrieval, the model is able to remove noise that would otherwise be ignored in the baseline RAG model. Further, by incorporating cosine similarity in addition to the FAISS search, the LDA+RAG model leverages two different similarity metrics that could isolate documents whose semantic content more closely aligns with the question.

It is interesting to note the disparity in the ranking of F_1 scores between the two models among the different samples: $s = 47$ results in the highest F_1 score for the LDA+RAG model, whereas $s = 1$ results in the highest F_1 score

for the baseline RAG model. This disparity could be due to the influence of k for the baseline RAG model. As k increases, the performance of the baseline RAG model becomes increasingly worse. The decrease in F_1 score from $s = 1$ ($k = 10$) to $s = 99$ ($k = 100$) is much larger than the decrease from $s = 1$ ($k = 10$) to $s = 47$ ($k = 50$). This accentuated drop-off in performance for the baseline RAG model could be because there is more noise associated with an increased number retrieved documents; the likelihood of retrieving irrelevant documents increases, which can reduce the signal from the provided context. However, the LDA+RAG model seems to avoid this steep drop-off in performance because the retrieved documents are more relevant (and the number of retrieved documents might even be less than k if the number of documents that pass the threshold is less than k), so the model is more likely to find the correct answer in the text.

In addition to comparing F_1 scores between models, I was curious to see whether including the LDA step reduced the amount of time needed to process each query. To test this question, I timed how long it took each model to process each query in each of the samples and calculated the average processing time per query. The results are shown below:

Table 6.3: Average time processing each query (in seconds)

	LDA+RAG	RAG
$s = 1$	**0.0169**	0.0795
$s = 47$	**0.0557**	0.3453
$s = 99$	**0.1025**	0.6747

We notice that for each sample, the LDA+RAG model has a substantially shorter average query processing time than the baseline RAG model. This makes sense because the search space of documents for retrieval is smaller after LDA step is performed. Observe that as k increases, the average query processing time increases, as more documents will be retrieved for each query, increasing the amount of time it takes to generate a response. However, it is important to note that this experiment was only focused on the query processing time. If we factor in the time taken to perform the LDA preprocessing step, the overall runtime for the LDA+RAG model would be longer than the runtime for the baseline RAG model.

6.2 Comparing Results to Base QA Model

Next, I compare the performance of the RAG-based models in the experiment to the *roberta-base-squad2* model (which I refer to as the Base QA model). In the previous section, I excluded results from the Base QA model because, unlike the RAG models, the Base QA model does not incorporate a retrieval component and is not fine-tuned on an external knowledge base. Instead, the only input the model sees is a question and context paragraph, from which the answer is chosen. Because the model does not incorporate nonparametric memory, there is no added "noise" from external documents (such as those in the *databricks-dolly-15k* dataset), so the model has a higher chance of extracting the right answer. As such, the Base QA model outperformed the RAG models in both accuracy and runtime, which is shown below:

Table 6.4: F_1 score

	LDA+RAG	RAG	Base QA
$s = 1$	84.64	81.19	**91.18**
$s = 47$	87.39	78.49	**92.16**
$s = 99$	83.19	69.52	**90.89**

We observe that the Base QA model outperforms the RAG models in F_1 score for every sample. The pattern of scores for the base model follows that of the LDA+RAG model: the scores from the $s = 47$ sample are highest, followed by those from $s = 1$, and then those from $s = 99$. This result stems from the fact that the Base QA model has a higher chance of extracting the correct answer because it only sees the provided context, whereas the RAG-based models sift through thousands of additional documents, introducing considerably more noise.

Table 6.5: Average time processing each query (in seconds)

	LDA+RAG	RAG	Base QA
$s = 1$	0.0169	0.0795	**0.0083**
$s = 47$	0.0557	0.3453	**0.0080**
$s = 99$	0.1025	0.6747	**0.0083**

The above table suggests that the Base QA model has a much faster average query processing time than each of the RAG-based models. This makes sense because the Base QA model only considers the context paragraph, whereas the RAG models iterate through hundreds, if not thousands, of additional documents. In addition, it is interesting to note that the Base QA model has a stable processing time regardless of sample; this differs from the RAG models, which have processing times that are dependent on the value of the *top-k* parameter.

6.3 Extending Results to Full SQuAD 2.0 Dataset

This section includes results from the model evaluated on the full *SQuAD 2.0* dataset. Recall that the full dataset includes entries in which the correct answer is an empty string, so it is much more difficult for the models to perform well because they all output some response (leading to an F_1 score of 0 for all such entries). As such, the average F_1 scores for each of the models were substantially worse than the F_1 scores shown earlier for the condensed dataset. With that being said, these results follow the same pattern as those previously, where the Base QA model performs best, the LDA+RAG model performs second-best, and the baseline RAG model performs worst for both F_1 score and average query processing time. The results are shown below:

Table 6.6: Best hyperparameter configurations for entire dataset

	num_topics	*threshold*	k	F_1 score
$s = 1$	20	0.5	50	41.61
$s = 47$	30	0.7	10	36.80
$s = 99$	30	0.5	50	46.39

Table 6.7: F_1 score for entire dataset

	LDA+RAG	RAG	Base QA
$s = 1$	41.61	35.37	**43.83**
$s = 47$	36.80	30.84	**37.97**
$s = 99$	46.39	41.99	**53.32**

Table 6.8: Average query processing time for entire dataset

	LDA+RAG	RAG	Base QA
$s = 1$	0.056	0.341	**0.0080**
$s = 47$	0.017	0.079	**0.0079**
$s = 99$	0.054	0.339	**0.0080**

Chapter 7

Discussion

The results from this experiment suggest that using LDA as a pre-processing step before retrieval leads to higher F_1 scores on average and faster processing time per query when compared to a baseline RAG model. As such, incorporating LDA in future RAG-based models seems like a plausible option for transforming a closed-book QA model into an open-domain QA model. Recall that in a closed-book QA approach, the model only sees the question and the provided context, which contains the answer. In an open-domain QA approach, the model considers a large corpus of text, which contains the answer. In this experiment, I effectively extend the capabilities of the *deepset/roberta-base-squad2* model from that of a closed-book QA model (which only sees the *SQuAD 2.0* dataset) to an open-domain QA model, which learns from the nonparametric *databricks-dolly-15k* dataset.

Although I was able to incorporate LDA to improve model performance, there were limitations in my experiment that placed a ceiling on how much I could improve the model. Many of these limitations revolved around time and computation constraints. For example, to choose the optimal hyperparameters for my model, I performed a grid search over only 27 possible permutations (3 options for each of the 3 parameters), so my current configuration of hyperparameters might not have been the most optimal. Had I had more time and access to computational resources, I might have performed a more extensive grid search or implemented cross-validation to select the optimal hyperparameters. In addition, I could have tuned the model on a more comprehensive and relevant external database such as *wiki_dpr*, which might have increased model performance even more. Lastly, rather than evaluating

my model on random samples of the abridged *SQuAD 2.0* dataset, I could have evaluated it on the entire abridged dataset.

As an open-domain QA model, there are a plethora of future directions that can improve model performance. One idea could be to not only train the model on a more comprehensive dataset but to also test this model on other benchmark evaluation datasets besides *SQuAD 2.0* such as *Natural Questions* and *TriviaQA*. A different method to improve model performance could be to prioritize the minimization of hallucinations by fine-tuning the model on datasets that provide references and testing how well the model utilizes these references. Lastly, and perhaps more feasibly, a logical progression from this experiment is to combine other topic-modeling algorithms such as NMF or LSA with RAG and compare performance against the LDA-based model.

Bibliography

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Conover, M., Hayes, M., Mathur, A., Xie, J., Wan, J., Shah, S., Ghodsi, A., Wendell, P., Zaharia, M., and Xin, R. (2023). Free dolly: Introducing the world's first truly open instruction-tuned llm.

Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L., and Jégou, H. (2024). The faiss library.

Griffiths, T. L. (2004). Finding scientific topics. *PNAS*.

Karpukhin, V., Oguz, B., Min, S., Lewis, P. S., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020). Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Mukherjee, A. (2016). Gibbs sampler derivation for latent dirichlet allocation. Technical report, University of Houston.

Rajpurkar, P., Jia, R., and Liang, P. (2018). Know what you don't know: Unanswerable questions for SQuAD. In Gurevych, I. and Miyao, Y., editors, *Proceedings of the 56th Annual Meeting of the Association for Compu-*

tational Linguistics (Volume 2: Short Papers), pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. (2020). Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788.