# DEVELOPING A SMART PERSONALIZED MAIL-TAGGER SYSTEM

***A B. Tech Project Report Submitted in Partial Fulfillment of the Requirements for the Degree of***

**Bachelor of Technology**

*by*

**Aditya Raj and Modukuri Venkata Nagasurya**
(112001003, 112001027)

*under the guidance of*

**Dr. Koninika Pal**

INDIAN INSTITUTE
OF TECHNOLOGY
PALAKKAD

**to the**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD**

# CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**DEVELOPING A SMART PERSONALIZED MAIL-TAGGER SYSTEM**" is a bonafide work of **Aditya Raj and Modukuri Venkata Nagasurya** (**Roll No. 112001003, 112001027**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Palakkad under my supervision and that it has not been submitted elsewhere for a degree.*

**Dr. Koninika Pal**
Assistant Professor,
Department of Computer Science & Engineering,
Indian Institute of Technology Palakkad, Kerala.

# Acknowledgements

We would like to express our sincere gratitude to our mentor Dr. Koninika Pal for providing her invaluable guidance, comments and suggestions throughout the course of the project.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In recent years, social media platforms like Twitter, Facebook, and Google+ have made short pieces of writing a popular way to share information online. These short texts can be anything from emails, news headlines and summaries of web pages to product ads, tweets, status updates, and even question-and-answer exchanges. Because they are so short, these texts can be tricky for computers to analyze using topic modelling techniques.

Topic modelling is a machine-learning approach that unveils hidden thematic structures within a collection of documents. However, traditional topic modelling methods were designed for analyzing longer documents and struggled with the sparse nature, high dimensionality, and sheer volume of short texts. This is why we need new and improved topic modelling methods specifically designed for short text analysis.

## 1.1 Objective

Our project aims to create a personalized email tag-generating system that helps users efficiently manage a high volume of emails and prioritize their inboxes. Prior research employing LDA on emails yielded unsatisfactory results due to the inherent shortcomings of the technique when applied to short text.

Latent Dirichlet Allocation (LDA) is a well-established topic modelling technique that excels at identifying thematic structures within large document collections. However, its effectiveness diminishes when applied to short texts.

The objective of this project is to tackle this challenge by incorporating a text-generation approach. This technique aims to augment short texts with additional content that faithfully preserves the original meaning and context. By transforming these short texts into more elaborate representations, we enable the application of conventional topic modelling methods like LDA. In this expanded format, LDA is expected to perform significantly better in uncovering the latent thematic structures within the corpus.

## 1.2 Problem Statement

We have a set of short text documents, denoted as $D = \{d_1, d_2, \ldots, d_n\}$. Each of these documents is characterized by a notably limited word count, which presents a significant hurdle in our quest to accurately assign topical labels to them. The inherent constraint here is that these documents lack the substantial word co-occurrence patterns which are typically required for conventional topic modelling techniques, such as Latent Dirichlet Allocation (LDA), to perform optimally.

To surmount this particular challenge, consider a model $M$ which extends each document $d_i \in D$, resulting in a new set of extended documents $D' = \{d'_1, d'_2, \ldots, d'_n\}$. The model then applies LDA on the entire document set $D'$, resulting in a set of latent topics (assume, the number of topics is $k$), denoted as $T = M(D') = \{t_1, t_2, \ldots, t_k\}$ and a probability distribution over the identified topics, represented as $P(d'_i) = \{p_{t_1}, p_{t_2}, \ldots, p_{t_k}\}$. Each of the documents $d'_i$ is mapped to a $\tau \subset T$ . We will take 3 or 4 labels for each document from the probability distribution with the highest probability value.

## 1.3 Organization of The Report

In the first chapter, we lay the groundwork by introducing our project, stating our objectives, and outlining the problem we aim to address. Chapter 2 focuses on previous work in the field and the limitations encountered.

Chapter 3 discusses the literature survey for Topic Generation for Short-text modelling. In Chapter 4, we discuss the proposed model. All the experiments and results with the visualisation are in Chapter 5.

**Note:** '*' in front of the chapter name represents the previous semester's work.

# Chapter 2

# * Review of Prior Work

This chapter revisits the foundational concepts employed throughout this work. We provide definitions for essential terms such as term frequency (TF), document frequency (DF), and inverse document frequency (IDF). Subsequently, we discuss topic modeling and the experimental results obtained by applying Latent Dirichlet Allocation (LDA) to our email dataset.

## 2.1 Background and Preliminaries

### 2.1.1 Term Frequency

*Term Frequency* is a measure of how frequently a term occurs within a given document. It is calculated as:

$$tf_{t,d} = \frac{n_{t,d}}{|D|} \tag{2.1}$$

where $n_{t,d}$ is the number of times a term $t$ occurs in document $d$ and $D$ is the document length.

### 2.1.2 Document Frequency

*Document Frequency* is a measure of how common a term is in a collection of documents. It is calculated by counting the number of documents in the collection that contain the term. It is calculated as:

$$df_d = n_{t,d} \tag{2.2}$$

where $n_{t,d}$ is the number of times a term $t$ occurs in document $d$.

### 2.1.3 Inverse Document Frequency

*Inverse Document Frequency (IDF)* is a measure of how important a term is to a document in a collection of documents. It is calculated as:

$$idf_t = \frac{N}{df_t} \tag{2.3}$$

where $N$ is the total number of documents in the collection and $df_t$ is the number of documents that contain the term.
When the value of $N$ is very large then the $IDF$ value can blow up. Often *log* of IDF is taken to dampen this effect.

$$idf_t = \log(\frac{N}{df_t}) \tag{2.4}$$

### 2.1.4 Term Frequency- Inverse Document Frequency

*Term Frequency- Inverse Document Frequency (TF-IDF)* is a statistical score which reflects how important a term is to a document in a corpus. It is calculated as:

$$tf - idf_{t,d} = tf_{t,d} * idf_t \tag{2.5}$$

The tf-idf value increases proportionally to the number of times a term appears in a document and is o set by the number of documents in the corpus containing the word, which will give more advantage to documents containing exotic terms more frequently.

## 2.2 Topic Models

*Topic models* are used to uncover the latent semantic structure from the text corpus. It is a statistical model for finding the abstract *topics* present in a collection of documents. This model considers each document as a set of topics, and each topic as a set of terms, or specifically, a cluster of similar terms. Topic models are probabilistic generative models, in the sense that they view documents as being generated from a probabilistic distribution of topics. Similarly, each topic is visualized as a probabilistic distribution over a fixed vocabulary.

### 2.2.1 Latent Dirichlet Allocation

This section explores Latent Dirichlet Allocation (LDA), a fundamental technique in topic modeling. LDA helps us discover hidden thematic structures within large document collections, also known as corpora. These hidden themes are what we refer to as "latent" topics, meaning they are not explicitly stated in the documents but can be uncovered through computational analysis.

The intriguing part of LDA lies in its use of a statistical concept called the Dirichlet distribution. This distribution acts like a mathematical model that governs how topics are distributed across documents and how words are distributed across topics. Imagine it as a set of rules that determines how frequently topics appear in documents and how often specific words are associated with those topics.

The "allocation" aspect of LDA refers to the process of assigning these hidden topics within individual documents. Essentially, LDA performs a two-step task:

- **Topic Discovery**: It identifies the underlying thematic structures (topics) within the entire document collection (corpus).

- **Document-Topic Distribution**: It determines the relevance of each topic to each document. This can be visualized as a "topic profile" for each document, indicating the degree to which it aligns with the discovered topics. Think of it as a way of measuring how much each document is "about" a particular topic.

Its worth highlighting that the ultimate objective of LDA is to unveil the distributions of topics across documents and terms within topics. In a substantial document collection, LDA diligently identifies groups of terms associated with specific topics. However, the task of assigning a meaningful label or interpretation to each topic is something that's left in the hands of the user, often demanding specialized knowledge. This essentially categorizes LDA as an unsupervised learning method. It excels at pattern discovery, but the naming part of the game is where human expertise comes into play. In a way, its like LDA sets the stage, but its up to us, with our domain knowledge, to put on the show and give those topics meaningful identities.

Both the document-topic and topic-word probabilities form Dirichlet distributions. The

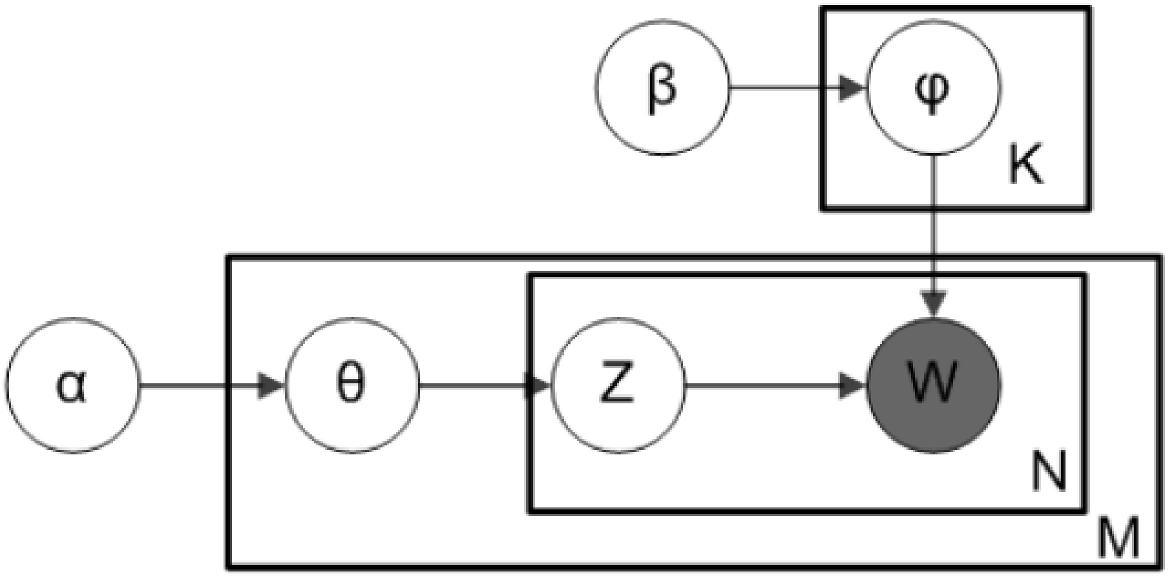


**Fig. 2.1** Latent Dirichlet Allocation Model [1]

LDA model has two Dirichlet parameters for controlling these distributions:

$\alpha$: controls the per-document topic distribution.

$\beta$: controls per-topic word distribution.

$M$: Number of documents.

$N$: Number of terms in a document.

$K$: Number of topics.

$Z$: The latent topic assigned to the word.

$W$: The specific word in the document.

$\theta$: Topic distribution for document.

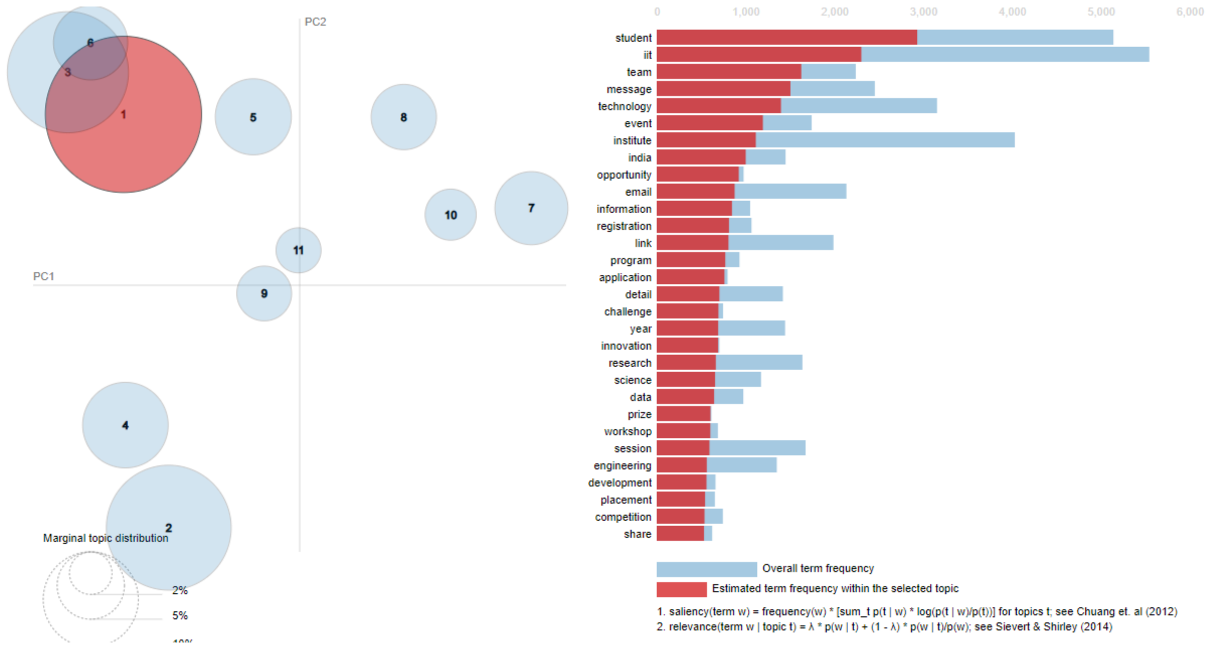
$\phi$: Word distribution for topic.

**Generative Process:** LDA assumes a generative process which can be explained as follows: for each word in the document, choose a topic and for each topic choose a term from the cluster for that topic. This generation is based on probability distributions.

1. Pick a document-topic distribution $\theta_i \sim Dir(\alpha)$ where $i \in \{1, \ldots, M\}$
2. Pick a topic-word distribution $\phi_k \sim Dir(\alpha)$ where $k \in \{1, \ldots, K\}$
3. For each word position $w_{i,j}$ where $i \in \{1, \ldots, M\}$ and $j \in \{1, \ldots, N_i\}$:
   - Pick a topic $z_{i,j} \sim Multinomial(\theta_i)$ (i.e. topic according to the document-topic distribution chosen).
   - Pick a word $w_{i,j} \sim Multinomial(\phi_{z_{i,j}})$ (i.e. word according to the topic-word distribution chosen).

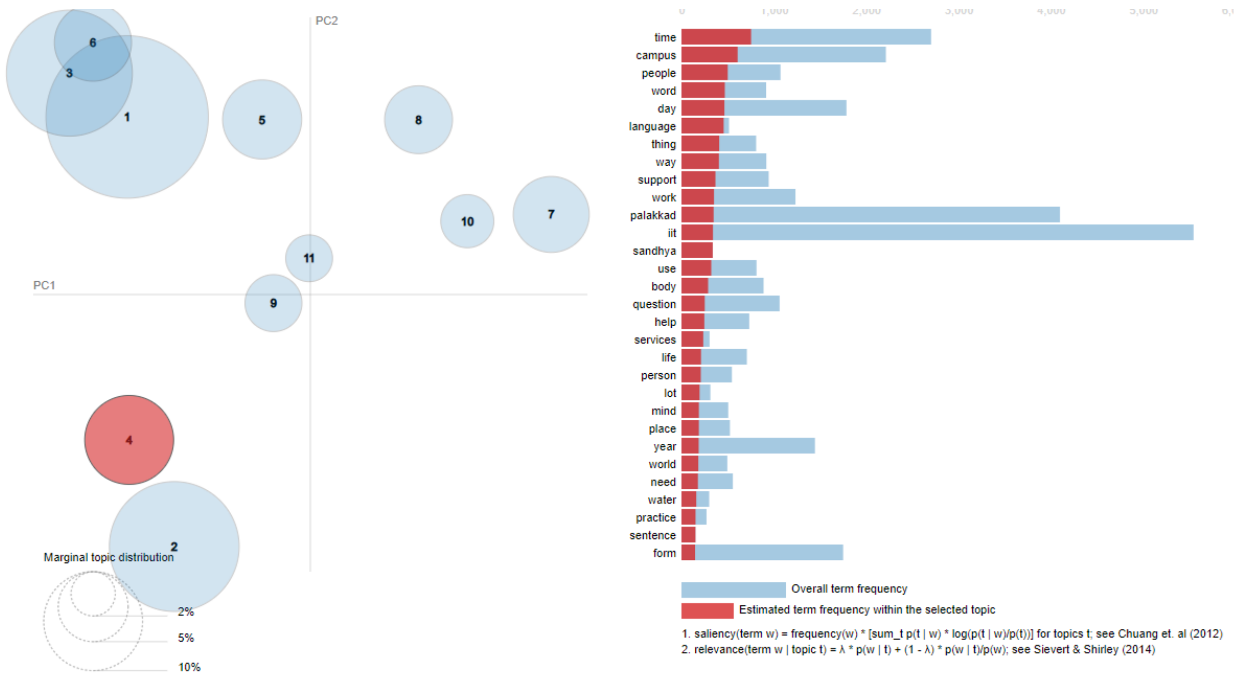
## 2.3 Experiment

We have extracted all our Gmail inbox messages as mbox, and ran the code done by the previous year students (which implemented LDA model for assigning topic labels to each email document) on our mbox. Then the topic clustering achieved through LDA is shown in figure 2.1 to figure 2.4.



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

**Fig. 2.2** Visualisation of LDA clustering- Cluster 1



1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

**Fig. 2.3** Visualisation of LDA clustering- Cluster 4

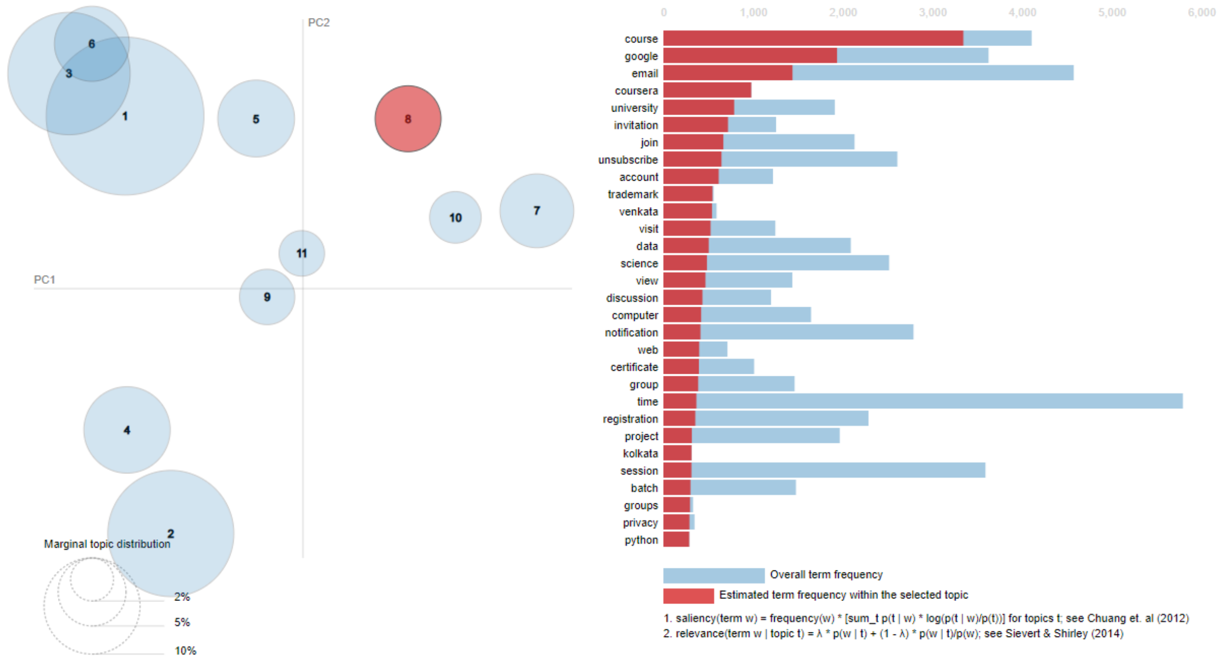


**Fig. 2.4** Visualisation of LDA clustering- Cluster 8

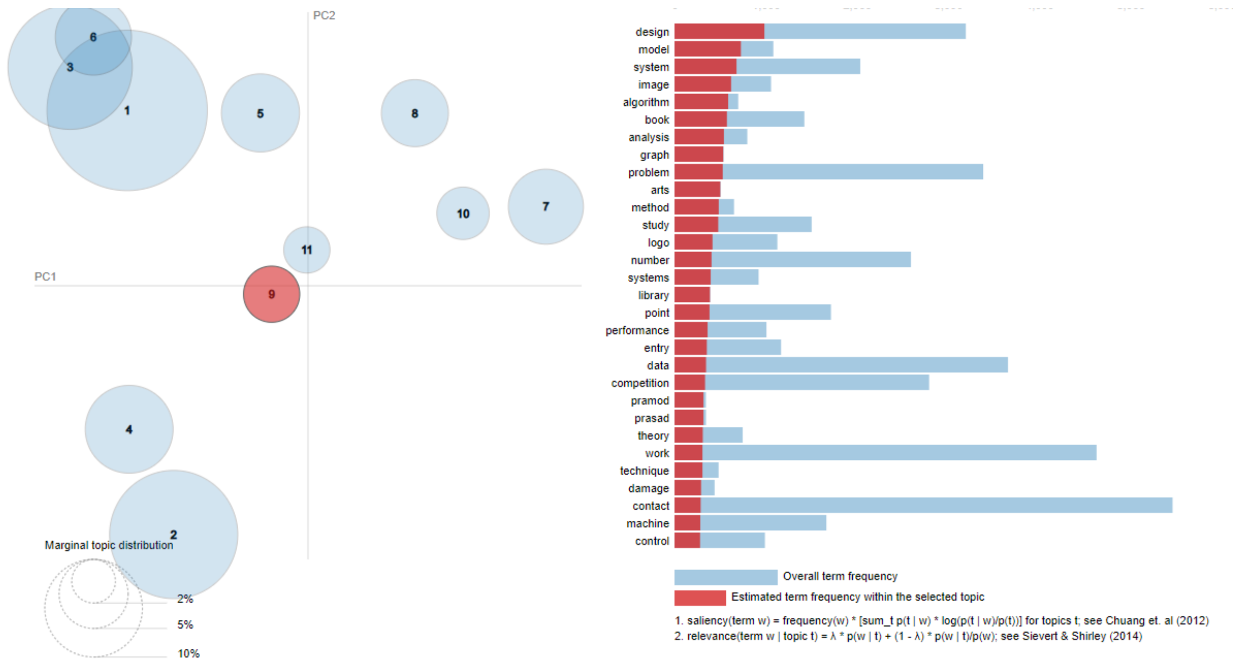


**Fig. 2.5** Visualisation of LDA clustering- Cluster 9

# Chapter 3

# * Literature Survey For Topic Generation For Short Text

## 3.1 Gibbs Sampling for Dirichlet Multinomial Mixture (GSDMM)

Gibbs Sampling for Dirichlet Multinomial Mixture (GSDMM) [2] is a topic modelling technique specifically designed for short text documents like emails.

### 3.1.1 Challenge

Traditional topic modelling techniques like Latent Dirichlet Allocation (LDA) struggle with short texts because they rely on frequent word co-occurrences to identify thematic structures. Short documents simply don't have enough words for these patterns to emerge clearly.

### 3.1.2 Approach

GSDMM addresses this challenge by introducing a two-step process:

1. **Document Assignment:** Instead of assuming documents belong to multiple topics,

GSDMM assumes each short document is dominated by a single topic.

2. **Iterative Refinement:** GSDMM employs a statistical method called Gibbs Sampling to iteratively refine topic assignments for each document.

### 3.1.3 Benefits of GSDMM

- **Handles Short Texts:** By assuming a single dominant topic per document, GSDMM works well with short content where word co-occurrence patterns are weak.
- **Iterative Refinement:** Gibbs Sampling allows the model to continuously improve topic assignments as it learns from the entire document collection.

### 3.1.4 Limitations of GSDMM

- **Oversimplification:** Assuming a single topic per document might be too restrictive for some emails with nuances.
- **Computational Cost:** Iterative sampling can be computationally expensive for very large datasets.

## 3.2 GPU-DMM

The Generalized Pólya Urn Dirichlet Multinomial Mixture Model (GPUDMM) [3] is a topic modelling technique that builds upon the strengths of the Dirichlet Multinomial Mixture Model (DMM) and incorporates concepts from the Pólya Urn scheme.

### 3.2.1 DMM Recap

- DMM is a topic modelling approach suited for short texts.
- It assumes each document is dominated by a single topic.

- It uses Dirichlet Multinomial distribution to model word occurrences within documents based on the assigned topic.

### 3.2.2 Limitations of DMM

- DMM doesn't explicitly consider the order of words within a document, which can be informative for topic identification.
- It might struggle to capture documents with thematic nuances or those containing words that belong to multiple topics.

### 3.2.3 Generalized Pólya Urn (GPU) for Topic Modelling

- The GPU is a statistical model that introduces ”balls” representing words and ”colors” representing topics.
- When generating a word for a document, the model considers:
  - The existing word distribution within the current topic (like a bag of words for the topic).
  - A ”global vocabulary urn” containing all words across all topics.
- This allows the model to incorporate both topic-specific word preferences and the broader vocabulary context.

### 3.2.4 Benefits of GPUDMM

- **Improved Topic Modeling:** By considering word order and broader vocabulary context, GPUDMM can potentially capture thematic nuances and handle documents with words belonging to multiple topics more effectively than DMM.
- **Flexibility:** The model can be adjusted to control the influence of topic-specific word preferences versus the global vocabulary.

## 3.3 Named Entity (NE) informed Corpus Embedding for Topic Modelling (NEiCE)

NEiCE [4] is a topic modeling technique specifically designed for short text documents. It tackles the limitations of a short text by utilizing Non-Negative Matrix Factorization (NMF). NMF works by decomposing the document-term matrix, representing the entire corpus, into two lower-dimensional matrices. The first matrix captures document representations across topics. The second captures topic representations over words. Once NMF is done finding these matrices, we can use them to find the main topics in all the documents.

### 3.3.1 Approach

NEiCE aims to surpass the limitations of the simple Bag-of-Words (BoW) matrix representation (document-term matrix) by incorporating semantic information. This is achieved by leveraging Named Entities (NEs) due to their inherent property of carrying more semantic weight compared to other words within short documents. By incorporating NEs, NEiCE can effectively identify key topics within short texts and uncover relationships between them. It follows a two-step process: **preprocessing** and **computation**.

**Preprocessing Step.** This stage identifies NEs within the documents and subsequently links them to their most likely corresponding entities on Wikipedia. This step assigns confidence scores to each identified NE.

**Computation Step.** In this step, NEs with high confidence scores(from Wikipedia linking) are assigned higher weights in the document-term matrix before topic modelling. This essentially makes NEs more influential in defining the topics. The new corpus representation is then used for topic modelling.

### 3.3.2 Benefits

The focus on NEs and the resulting enriched corpus representation allow NEiCE to identify more accurate and meaningful topics compared to traditional topic modelling methods that treat all words equally.

### 3.3.3 Limitations

NEiCE works well for public figures and events documented on Wikipedia, but it can be problematic for short texts like institute emails. Often, these emails contain important names specific to the institute, such as professors or events, that might not be listed on Wikipedia. In these cases, NEiCE may not perform well.

# Chapter 4

# Proposed Topic Generation Model for Short Text Documents

Due to the shortcomings of the topic-generation models discussed in the previous chapter, we're exploring an alternative approach. This new method leverages pre-trained Large Language Models (LLMs) to first expand the short text. We'll then filter the most relevant expansion and feed it into a conventional topic modelling algorithm for topic identification. This two-step approach aims to improve accuracy by providing richer content for the topic model to analyze.

To achieve this, we leverage a powerful text generation model called LLaMA [5]. This model acts like a creative writer, and we use it to generate new sentences that are relevant to the short text document. While other options were considered, such as Google API Client's Custom Search [6] for finding relevant web pages and GPT-3.5-Turbo API [7] for additional text generation, we ultimately chose LLaMA for text generation.

It's important to note that not all generated sentences will be relevant. The method for selecting sentences that best match the context of the short text will be discussed in section 4.2. In section 4.1, the procedure followed for generating sentences using GPT, Google API and LLaMA model is discussed. Consider the short-text $s =$ "age variations cortical membranes" which will be used as an example in the following sections.

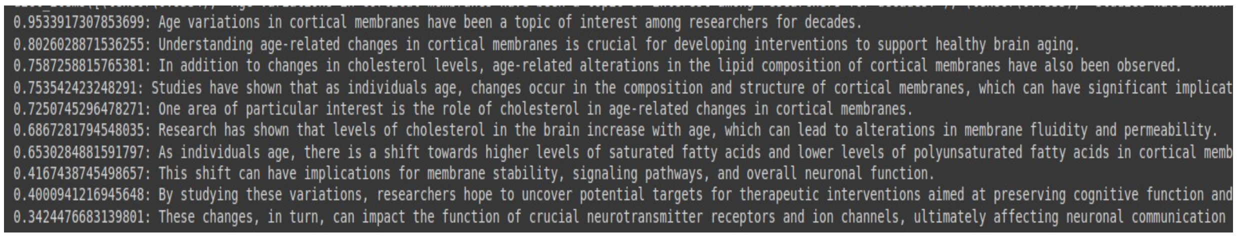
## 4.1 Text Generation

### 4.1.1 GPT API

To generate text relevant to the short text document, we leverage the GPT-3.5-Turbo model exposed through the GPT API. This large language model acts as a text generation engine, capable of producing new sentences that maintain the context of the original short text. We achieve this by providing a prompt to the GPT API. The generated text is then processed using the Natural Language Toolkit (NLTK) library to segment it into individual sentences. The prompt given to GPT API is:

```
{short-text} is a short-text document. Extend this short text with the
same context.
```

The Generated sentences (using GPT API) and their corresponding similarity scores with the short-text $s$ are shown in Figure 4.1.



```
0.9533917307853699: Age variations in cortical membranes have been a topic of interest among researchers for decades.
0.8026028871536255: Understanding age-related changes in cortical membranes is crucial for developing interventions to support healthy brain aging.
0.7587258815765381: In addition to changes in cholesterol levels, age-related alterations in the lipid composition of cortical membranes have also been observed.
0.753542423248291: Studies have shown that as individuals age, changes occur in the composition and structure of cortical membranes, which can have significant implicat
0.7250745296478271: One area of particular interest is the role of cholesterol in age-related changes in cortical membranes.
0.6867281794548035: Research has shown that levels of cholesterol in the brain increase with age, which can lead to alterations in membrane fluidity and permeability.
0.6530284881591797: As individuals age, there is a shift towards higher levels of saturated fatty acids and lower levels of polyunsaturated fatty acids in cortical memb
0.4167438745498657: This shift can have implications for membrane stability, signaling pathways, and overall neuronal function.
0.4000941216945648: By studying these variations, researchers hope to uncover potential targets for therapeutic interventions aimed at preserving cognitive function and
0.3424476683139801: These changes, in turn, can impact the function of crucial neurotransmitter receptors and ion channels, ultimately affecting neuronal communication
```

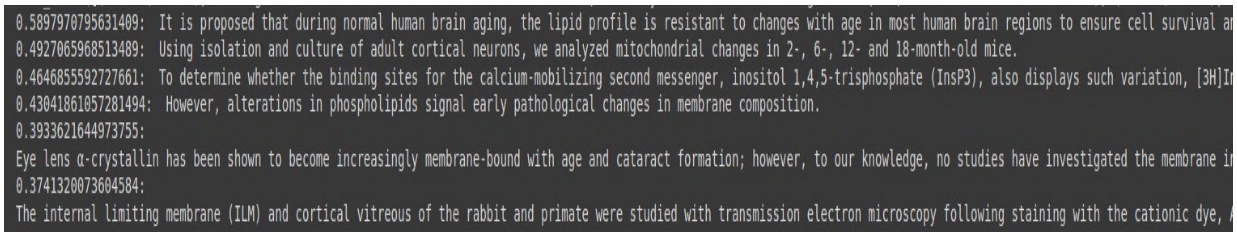
**Fig. 4.1** Generated Sentence along with similarity accuracy using GPT API

### 4.1.2 Google API

Google Custom Search API is another technique for text generation. This API allows us to create a custom search engine focused on retrieving webpages relevant to the short text under analysis. We then use the short text itself as a search query within this custom engine. This retrieves a list of URLs along with snippets.

We get the entire text from the URLs by web scraping the web pages. Since finding the keywords of the query within the entire text is computationally expensive. So, instead of searching for the keywords directly within the entire scraped text. We get the positions of these snippets within the scraped text. This allows us to directly extract the full sentences containing those snippets. These extracted sentences are likely to be highly relevant to the original short text and provide valuable context for further analysis.

The Generated sentences (using Google API) and their corresponding similarity scores with the short-text $s$ are shown in Figure 4.2.



```
0.5897970795631409:  It is proposed that during normal human brain aging, the lipid profile is resistant to changes with age in most human brain regions to ensure cell survival a
0.4927065968513489:  Using isolation and culture of adult cortical neurons, we analyzed mitochondrial changes in 2-, 6-, 12- and 18-month-old mice.
0.4646855592727661:  To determine whether the binding sites for the calcium-mobilizing second messenger, inositol 1,4,5-trisphosphate (InsP3), also displays such variation, [3H]I
0.43041861057281494:  However, alterations in phospholipids signal early pathological changes in membrane composition.
0.3933621644973755:
Eye lens α-crystallin has been shown to become increasingly membrane-bound with age and cataract formation; however, to our knowledge, no studies have investigated the membrane i
0.3741320073604584:
The internal limiting membrane (ILM) and cortical vitreous of the rabbit and primate were studied with transmission electron microscopy following staining with the cationic dye,
```

**Fig. 4.2** Generated Sentence along with similarity accuracy using Google API

### 4.1.3 LLaMA (Large Language Model for Anything)

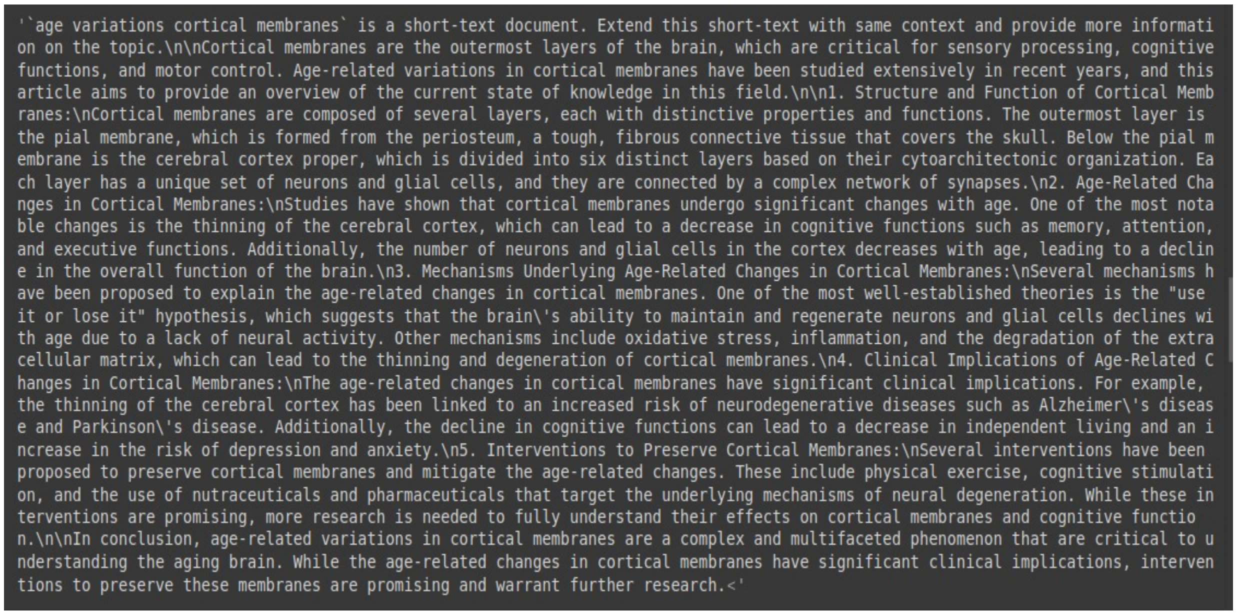
LLaMA (Large Language Model for Anything) is a large language model (LLM) developed by Meta AI (formerly Facebook AI). It is a type of artificial intelligence (AI) model trained on a massive dataset of text and code. It can generate text, translate languages, write different kinds of creative content, and answer your questions in an informative way. Compared to other LLMs, LLaMA emphasizes efficiency by focusing on training data volume rather than just the number of model parameters. This can potentially lead to lower computational costs during training and inference.

This large language model acts as a text generation engine, capable of producing new sentences that maintain the context of the original short text. We achieve this by providing a prompt to the LLaMA. The generated text is then processed using the Natural Language

Toolkit (NLTK) library to segment it into individual sentences. The prompt given to LLaMA is:

```
{short-text} is a short-text document. Extend this short text with the
same context.
```

The Generated sentences (using LLaMA model) with the short-text $s$ are shown in Figure 4.3.



```
'`age variations cortical membranes` is a short-text document. Extend this short-text with same context and provide more informati
on on the topic.\n\nCortical membranes are the outermost layers of the brain, which are critical for sensory processing, cognitive
functions, and motor control. Age-related variations in cortical membranes have been studied extensively in recent years, and this
article aims to provide an overview of the current state of knowledge in this field.\n\n1. Structure and Function of Cortical Memb
ranes:\nCortical membranes are composed of several layers, each with distinctive properties and functions. The outermost layer is
the pial membrane, which is formed from the periosteum, a tough, fibrous connective tissue that covers the skull. Below the pial m
embrane is the cerebral cortex proper, which is divided into six distinct layers based on their cytoarchitectonic organization. Ea
ch layer has a unique set of neurons and glial cells, and they are connected by a complex network of synapses.\n2. Age-Related Cha
nges in Cortical Membranes:\nStudies have shown that cortical membranes undergo significant changes with age. One of the most nota
ble changes is the thinning of the cerebral cortex, which can lead to a decrease in cognitive functions such as memory, attention,
and executive functions. Additionally, the number of neurons and glial cells in the cortex decreases with age, leading to a declin
e in the overall function of the brain.\n3. Mechanisms Underlying Age-Related Changes in Cortical Membranes:\nSeveral mechanisms h
ave been proposed to explain the age-related changes in cortical membranes. One of the most well-established theories is the "use
it or lose it" hypothesis, which suggests that the brain\'s ability to maintain and regenerate neurons and glial cells declines wi
th age due to a lack of neural activity. Other mechanisms include oxidative stress, inflammation, and the degradation of the extra
cellular matrix, which can lead to the thinning and degeneration of cortical membranes.\n4. Clinical Implications of Age-Related C
hanges in Cortical Membranes:\nThe age-related changes in cortical membranes have significant clinical implications. For example,
the thinning of the cerebral cortex has been linked to an increased risk of neurodegenerative diseases such as Alzheimer\'s diseas
e and Parkinson\'s disease. Additionally, the decline in cognitive functions can lead to a decrease in independent living and an i
ncrease in the risk of depression and anxiety.\n5. Interventions to Preserve Cortical Membranes:\nSeveral interventions have been
proposed to preserve cortical membranes and mitigate the age-related changes. These include physical exercise, cognitive stimulati
on, and the use of nutraceuticals and pharmaceuticals that target the underlying mechanisms of neural degeneration. While these in
terventions are promising, more research is needed to fully understand their effects on cortical membranes and cognitive functio
n.\n\nIn conclusion, age-related variations in cortical membranes are a complex and multifaceted phenomenon that are critical to u
nderstanding the aging brain. While the age-related changes in cortical membranes have significant clinical implications, interven
tions to preserve these membranes are promising and warrant further research.<'
```

**Fig. 4.3** Generated Sentence using LLaMA model

## 4.2 Expansion of Short-Text

For text generation, we have decided to go with LLaMA. After employing the text generation technique described earlier, we have a collection of sentences potentially similar to the original short text. Our goal is to strategically append these sentences to the original text, enriching its content for topic modelling. This section explores three distinct algorithms designed to strategically integrate the generated text with the original content. We will delve into the details of each algorithm below.

### 4.2.1 Algorithm-I

Algorithm 1 employs a similarity-based selection strategy for integrating generated text with the original content. It calculates a similarity score between each generated sentence and the original text using the Sbert model. These scores are then utilized to construct a key-value dictionary, where the key represents the similarity score and the value is the corresponding generated sentence. Subsequently, the dictionary is sorted in descending order based on the similarity scores, effectively prioritizing sentences with higher thematic or semantic alignment. Finally, a predefined threshold is applied to filter the generated sentences. Only sentences exceeding this threshold are appended to the original text.

### 4.2.2 Algorithm-II

Algorithm II builds upon Algorithm I by implementing a cascaded similarity filtering approach. While Algorithm I directly appends sentences exceeding a predefined threshold to the original text, Algorithm II introduces an iterative process. Following the initial high-similarity appending from Algorithm 1, Algorithm 2 recalculates similarity scores. This time, the scores are calculated between the remaining generated sentences and the newly formed text (original text + appended high-similarity sentences). This process of appending, recalculating similarity, and threshold filtering continues iteratively until the first encountered sentence in the remaining pool falls below the predefined threshold.

### 4.2.3 Algorithm-III

Algorithm III adopts a greedy selection strategy similar to Algorithm II but with a focus on prioritizing the single most relevant sentence at each iteration. It leverages a cascaded similarity filtering approach akin to Algorithm II. However, unlike Algorithm II, which appends all sentences exceeding the threshold, Algorithm III prioritizes the single sentence with the highest similarity score after each recalculation.

**Algorithm 1** Extension of Short-Text using Algorithm-I

```
    Input $S = \{s_1, s_2, \ldots, s_n\}$, $Text$, $\theta_{alg}$
    Output $extendedText$
1: procedure EXTENSION($S$, $Text$, $\theta_{alg}$)
2:     $sentenceSimilarityMap \leftarrow dict\{\}$
3:     $extendedText \leftarrow Text$
4:
5:     for each sentence $s \in$ S do
6:         $sim \leftarrow similarity\_value(s, Text)$
7:         $sentenceSimilarityMap[sim] \leftarrow s$
8:     end for
9:
10:    sort $sentenceSimilarityMap$                    ▷ In Descending Order
11:
12:    for similarityScore, sentence in sentenceSimilarityMap do
13:        if similarityScore $>=$ $\theta_{alg}$ then
14:            $extendedText$ ←extendedText + $sentence$
15:        end if
16:    end for
17: end procedure
```

**Algorithm 2** Pseudo algorithm for Algorithm-II and Algorithm-III

```
    Input $S = \{s_1, s_2, \ldots, s_n\}$, $Text$, $\theta_{alg}$
    Output $extendedText$
 1: procedure Extension($S$, $Text$, $\theta_{alg}$)
 2:     $sentenceSimilarityMap \leftarrow \{\}$
 3:     $extendedText \leftarrow Text$
 4:
 5:     for each sentence $s \in$ S do
 6:         $sim \leftarrow similarity\_value(s, Text)$
 7:         $sentenceSimilarityMap[sim] \leftarrow s$
 8:     end for
 9:
10:     sort $sentenceSimilarityMap$                          ▷ In Descending Order
11:
12:     while True do
13:         $sentenceSimilarityMap1, extendedText, \theta_{alg} \leftarrow$ AppendText($T$, sentenceSimilarityMap, $\theta_{alg}$)
14:         if $\theta_{alg}$ == -1 then
15:             break
16:         end if
17:         $Text \leftarrow extendedText$
18:         $sentenceSimilarityMap \leftarrow sentenceSimilarityMap1$
19:     end while
20: end procedure
```

**Algorithm 3** Helper function used in Algorithm-II

```
1: function AppendText(Text, sentenceSimilarityMap, $\theta_{alg}$)
2:     $sentenceSimilarityMap1 \leftarrow dict\{\}$
3:     $unappendedSentences \leftarrow list[]$
4:     $ct \leftarrow 0$
5:     $extendedText \leftarrow Text$
6:     for each similarityScore, sentences $\in$ sentenceSimilarityMap do
7:         if similarityScore $>= \theta_{alg}$ then
8:             $extendedText \leftarrow$extendedText + $sentences$
9:             $ct \leftarrow$ct+1
10:            continue
11:        end if
12:        if ct == 0 then
13:            Return $sentenceSimilarityMap, Text, -1$
14:        else
15:            $unappendedSentences \leftarrow unappendedSentences + sentences$
16:        end if
17:    end for
18:    $n \leftarrow lengthofunappendedText$
19:    for each sentence $s \in$ unappendedSentences do
20:        $sim \leftarrow similarity\_value(s, Text)$
21:        $sentenceSimilarityMap1[sim] \leftarrow s$
22:    end for
23:
24:    sort $sentenceSimilarityMap1$                    ▷ In Descending Order
25:
26:    if Length(sentenceSimilarityMap1) == 0 then
27:        Return $sentenceSimilarityMap, extendedText, -1$
28:    end if
29:    Return $sentenceSimilarityMap1, extendedText, \theta_{alg}$
30: end function
```

**Algorithm 4** Helper function used in Algorithm-III

```
1: function AppendText(Text, sentenceSimilarityMap, $\theta_{alg}$)
2:     $sentenceSimilarityMap1 \leftarrow dict\{\}$
3:     $unappendedSentences \leftarrow list[]$
4:     $ct \leftarrow 0$
5:     $extendedText \leftarrow Text$
6:     for each similarityScore, sentence $\in$ sentenceSimilarityMap do
7:         if similarityScore $>= \theta_{alg}$ and ct == 0 then
8:             $extendedText$ $\leftarrow$extendedText + $sentence$
9:             $ct$ $\leftarrow$ct+1
10:            continue
11:        end if
12:        if ct == 0 then
13:            Return $sentenceSimilarityMap, Text, -1$
14:        else
15:            $unappendedSentences \leftarrow unappendedSentences + sentence$
16:        end if
17:    end for
18:    $n \leftarrow lengthofunappendedText$
19:    for each sentence $s \in$ unappendedSentences do
20:        $sim \leftarrow similarity\_value(s, Text)$
21:        $sentenceSimilarityMap1[sim] \leftarrow s$
22:    end for
23:
24:    sort $sentenceSimilarityMap1$                    ▷ In Descending Order
25:
26:    if Length(sentenceSimilarityMap1) == 0 then
27:        Return $sentenceSimilarityMap, extendedText, -1$
28:    end if
29:    Return $sentenceSimilarityMap1, extendedText, \theta_{alg}$
30: end function
```

# Chapter 5

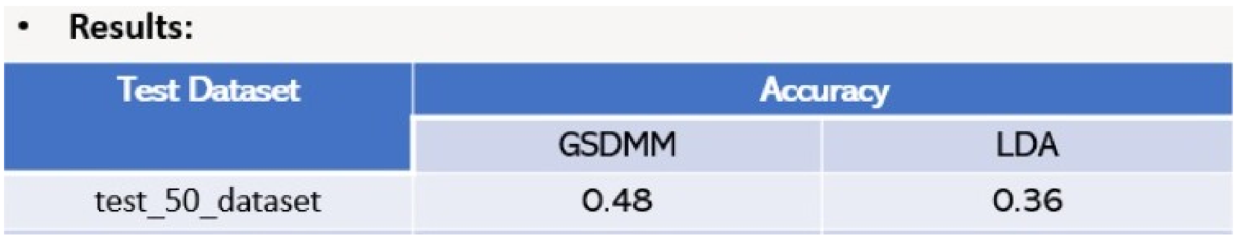
# Experimentation and Results

Our evaluation process utilized two distinct datasets. The first dataset consisted of emails extracted from our Google account mailbox (mbox format). This dataset comprised a total of 501 emails, with 451 designated for training the model and 50 reserved for testing purposes.

The second dataset leveraged for evaluation is a publicly available benchmark known as TagMyNews [8]. This dataset contains a vast collection of news article headlines. While extensive, we employed a subset of 1,000 short texts (headlines) for the training phase. The creation of two gold sets specifically for this dataset will be elaborated upon in Section 5.3.

## 5.1 Results of Baseline Models on Email Dataset

We evaluated the performance of the baseline models, namely GSDMM and LDA, on the email dataset. The training process utilized a dataset of 451 emails, while a separate set of 50 emails (referred to as the "test_50_dataset") was used for testing purposes. These 50 test emails were manually labelled, forming a Goldset. The resulting similarity accuracy (discussed in Section 5.2.1) achieved by these models on the test_50_dataset is illustrated in Figure 5.1.



- **Results:**

| Test Dataset | Accuracy | |
|---|---|---|
| | GSDMM | LDA |
| test_50_dataset | 0.48 | 0.36 |

**Fig. 5.1** Comparison of Similarity Accuracy

## 5.2 Evaluation Metrics

For evaluation, we compare the list of tags (predictedTags) predicted by models with the labels in the Goldset. The evaluation metrics discussed in Section 5.2.1 are used to evaluate the model.

### 5.2.1 Similarity Accuracy

This evaluation function utilizes a similarity metric based on GloVe word embeddings to assess the predicted tags. It marks a prediction as correct if the cosine similarity between at least one pair of predicted and expected tags exceeds a predefined threshold. We conducted our experiment using three different threshold values, which we will refer to as "$\theta_{sim}$" throughout this paper: 0.4, 0.45, and 0.5. The corresponding algorithm is shown in *Algorithm 5.*

## 5.3 Analysis and Visualization of Results for the TagMyNews Dataset

Our evaluation process began with extracting news headlines (short-text documents) from the TagMyNews dataset. We then employed LLaMA to generate new sentences that thematically aligned with these short texts. To ensure contextual coherence, we applied the algorithms discussed in Section 4.2. These algorithms filtered the generated sentences, selecting only those that exceeded a predefined threshold (which we will refer to as "$\theta_{alg}$"

**Algorithm 5** Similarity Accuracy

```
1: $correctPredictions \leftarrow 0$
2: for document in dataset do
3:     $predictedTags \leftarrow modelPrediction(document)$
4:     $similarity \leftarrow 0$
5:     for word1 in expectedTags do
6:         for word2 in predictedTags do
7:             if cosineSimilarity(word1, word2) $> \theta_{sim}$ then
8:                 $similarity \leftarrow similarity + 1$
9:             end if
10:        end for
11:    end for
12:    if similarity $> 1$ then
13:        $correctPredictions \leftarrow correctPredictions + 1$
14:    end if
15: end for
16: $similarityAccuracy \leftarrow correctPredictions/len(dataset)$
```

throughout this paper) of thematic similarity to the original short-text document.

To evaluate the performance of our algorithms, we created two distinct Goldsets. The first Goldset comprised 50 unseen short texts manually labelled with 4 tags for each short text. The second Goldset leveraged pre-existing labels(1 tag for each short text) within TagMyNews, involving 100 unseen short texts. These Goldsets will then assess how well our models predict labels for each short text.

To analyze and visualize the results, we employed different models (LDA, LDA+Glove and LDA+Sbert) for predicting thematic labels (tags) after applying our algorithms to short-text documents. Each model outputs the most relevant cluster for a given document. This cluster contains a probability distribution of topics over words. LDA directly assigns tags by selecting the top four most probable words within the identified cluster. LDA+Glove, which incorporates word embeddings, takes a two-step approach. First, it identifies the ten words with the highest probabilities from the best cluster. Subsequently, it utilizes word embedding similarity to find the four words with the highest similarity scores. These four words are then assigned as the predicted tags. We employed a similar strategy with LDA+Sbert, but utilized Sbert for the word embedding step.

### 5.3.1 Statistics of Algorithm I

The statistics of the number of sentences appended to each short-text document based on the chosen threshold value $\theta_{alg}$ are shown in Fig. 5.2.

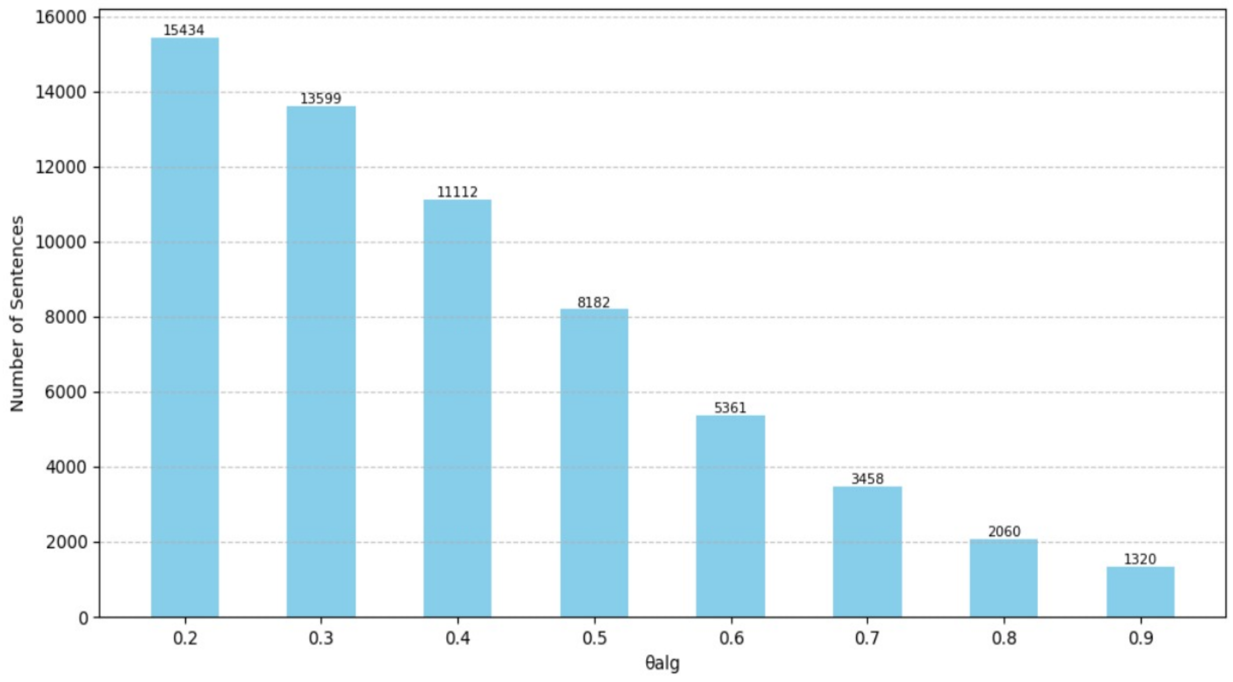


**Fig. 5.2** Number of Sentences Appended vs $\theta_{alg}$ (Algorithm I)

The accuracy of various models (LDA, LDA+Glove, and LDA+Sbert) after using algorithm I on short text for different values of "$\theta_{alg}$" (ranging from 0 to 1) when "$\theta_{sim}$" is set to 0.4 is illustrated in Figure 5.3.

### 5.3.2 Statistics of Algorithm II

#### Using the Pre-existing labelled Test Dataset

The statistics of the number of sentences appended to each short-text document based on the chosen threshold value $\theta_{alg}$ are shown in Fig. 5.4. The accuracy of models(LDA, LDA+Glove and LDA+Sbert) with various $\theta_{alg}$ values and $\theta_{sim}$ as 0.4, 0.45, 0.5 is illustrated in Figure 5.5, 5.6, 5.7 respectively.

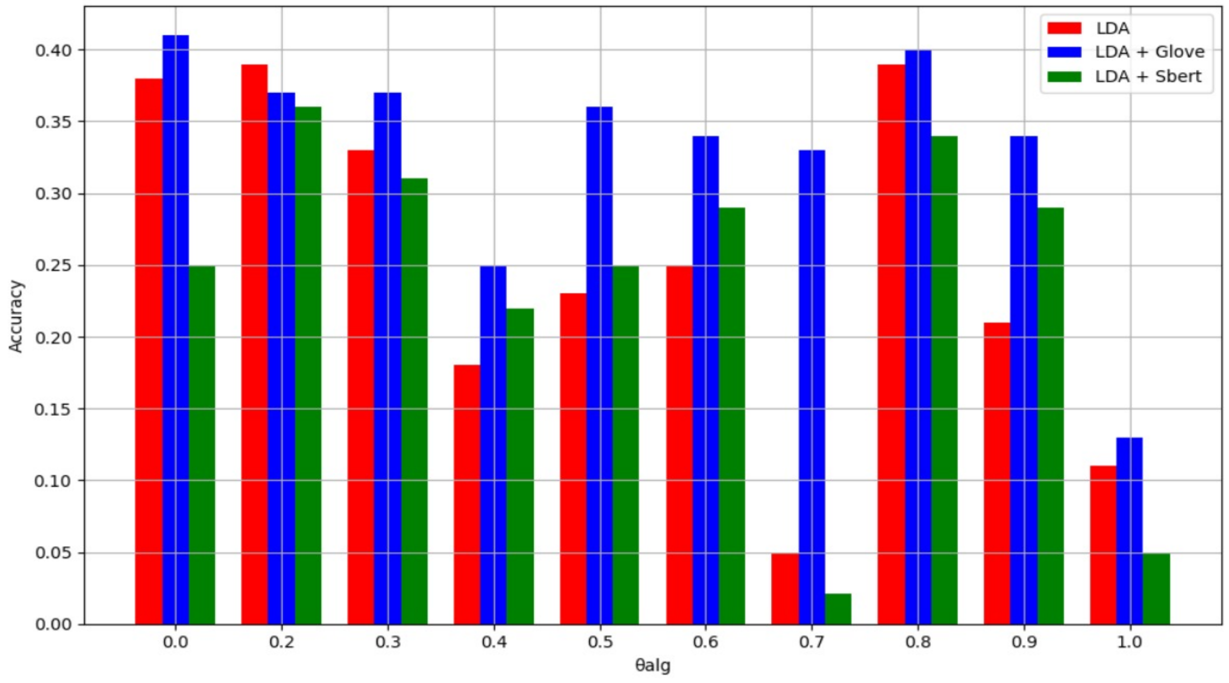


**Fig. 5.3** Accuracy of the Model using Algorithm I on various $\theta_{alg}$ and $\theta_{sim}$=0.4

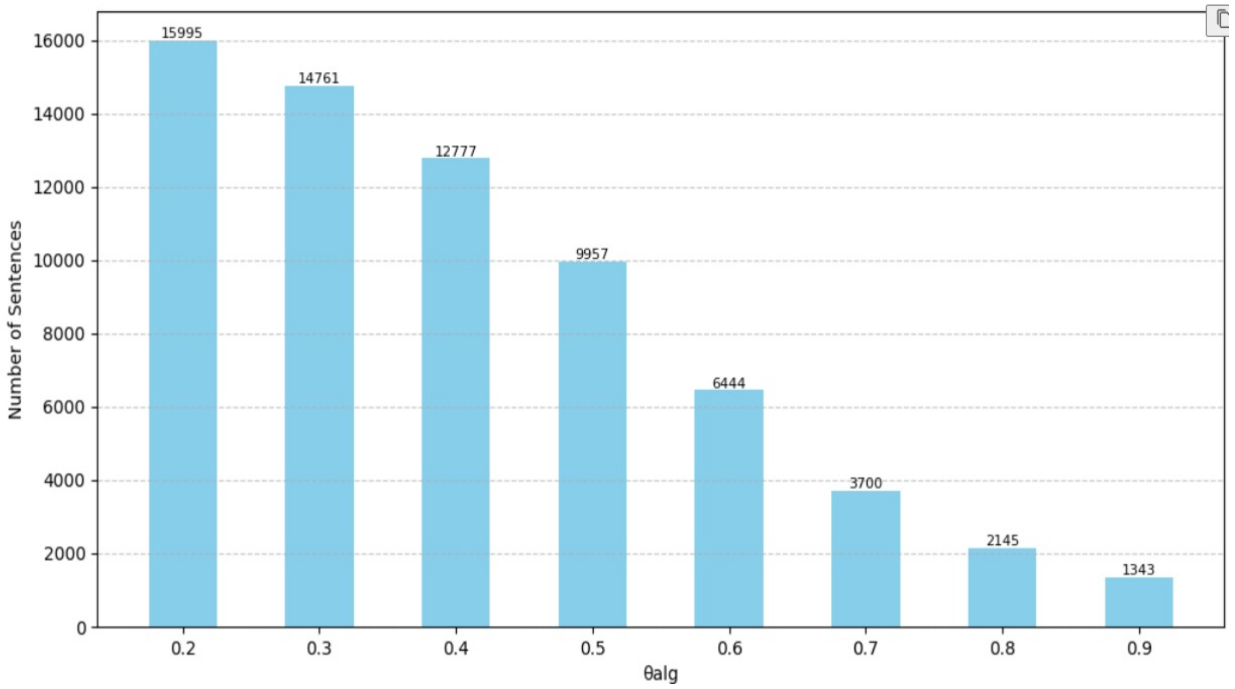


**Fig. 5.4** Number of Sentences Appended vs $\theta_{alg}$ (Algorithm II)

**Using Manually labelled Test Dataset**

The accuracy of models(LDA, LDA+Glove and LDA+Sbert) with $\theta_{alg}$ as 0.5 and $\theta_{sim} = \{0.4, 0.45, 0.5\}$ is illustrated in Figure 5.8.

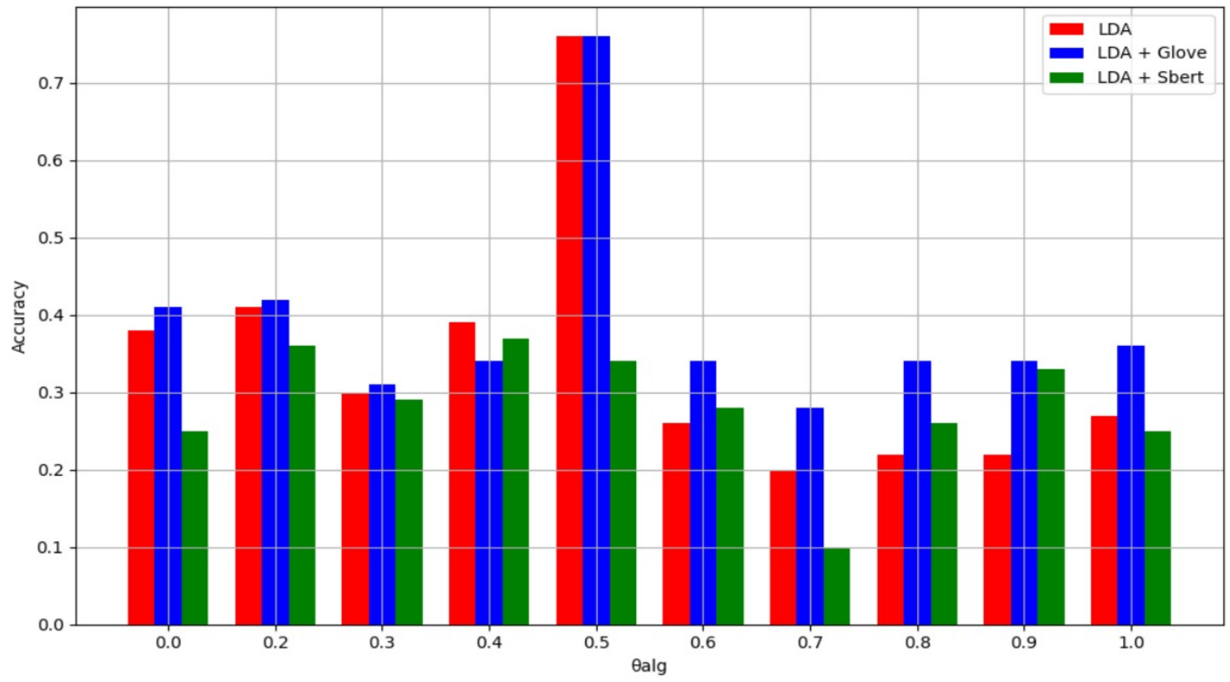


**Fig. 5.5** Accuracy of the Model using Algorithm II on various $\theta_{alg}$ and $\theta_{sim}$=0.4

### 5.3.3 Statistics of Algorithm III

When evaluating different values for "$\theta_{alg}$" in Algorithm II, we observed peak points within a specific range. This suggests that setting "$\theta_{alg}$" too low results in excessive noise being appended (irrelevant content) while setting it too high leads to very little content being appended. We identified a mid-range value (around 0.5) that offered a balance between minimizing noise and incorporating a reasonable amount of relevant content. Based on this observation, we decided to focus further evaluation on Algorithm III using this optimal "$\theta_{alg}$" value (0.5), while exploring different settings for "$\theta_{sim}$" as shown in Fig. 5.9.

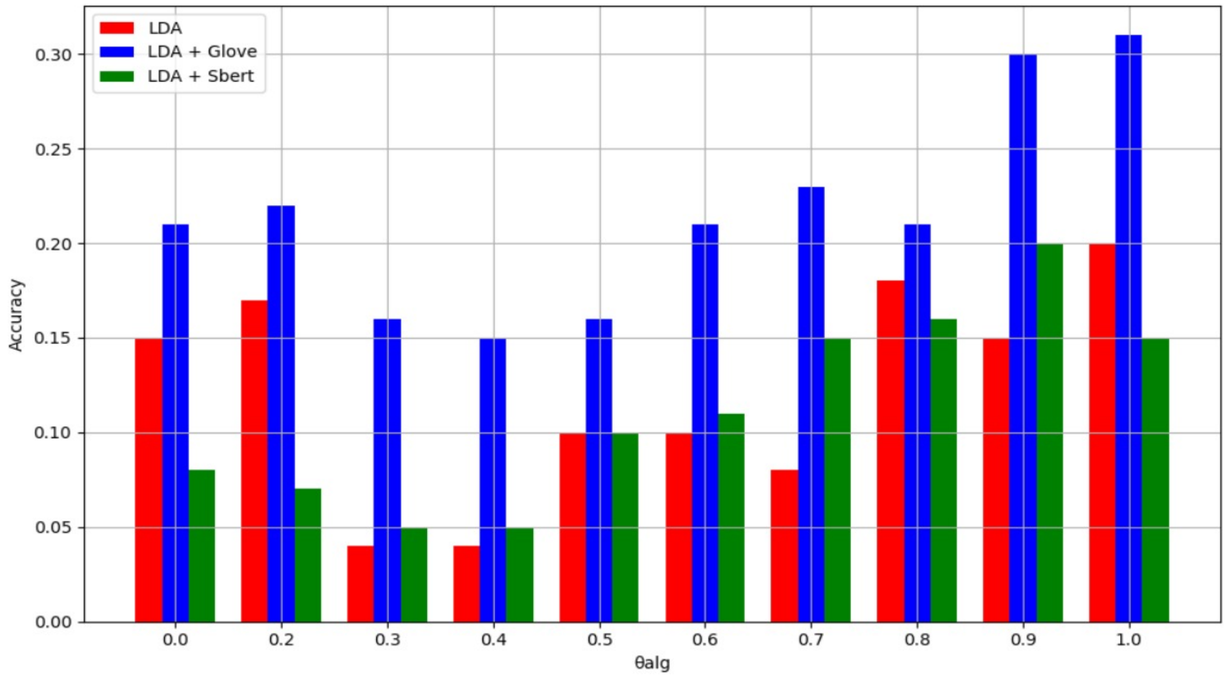


**Fig. 5.6** Accuracy of the Model using Algorithm II on various $\theta_{alg}$ and $\theta_{sim}$=0.45

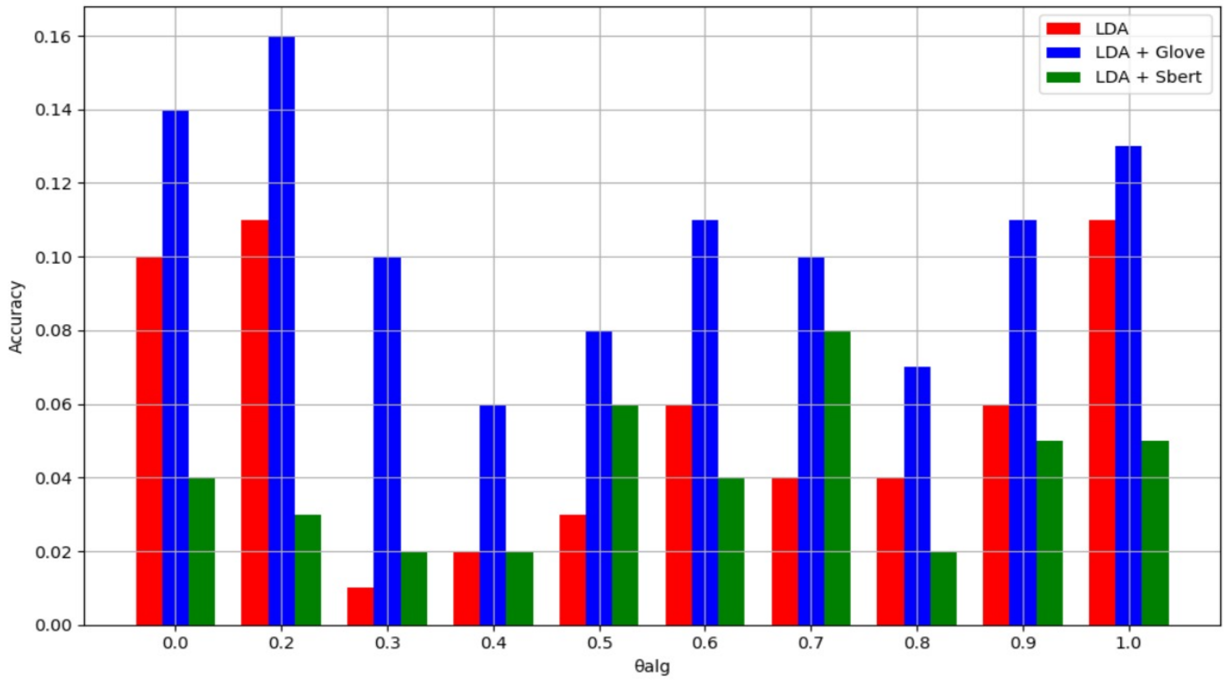


**Fig. 5.7** Accuracy of the Model using Algorithm II on various $\theta_{alg}$ and $\theta_{sim}$=0.5

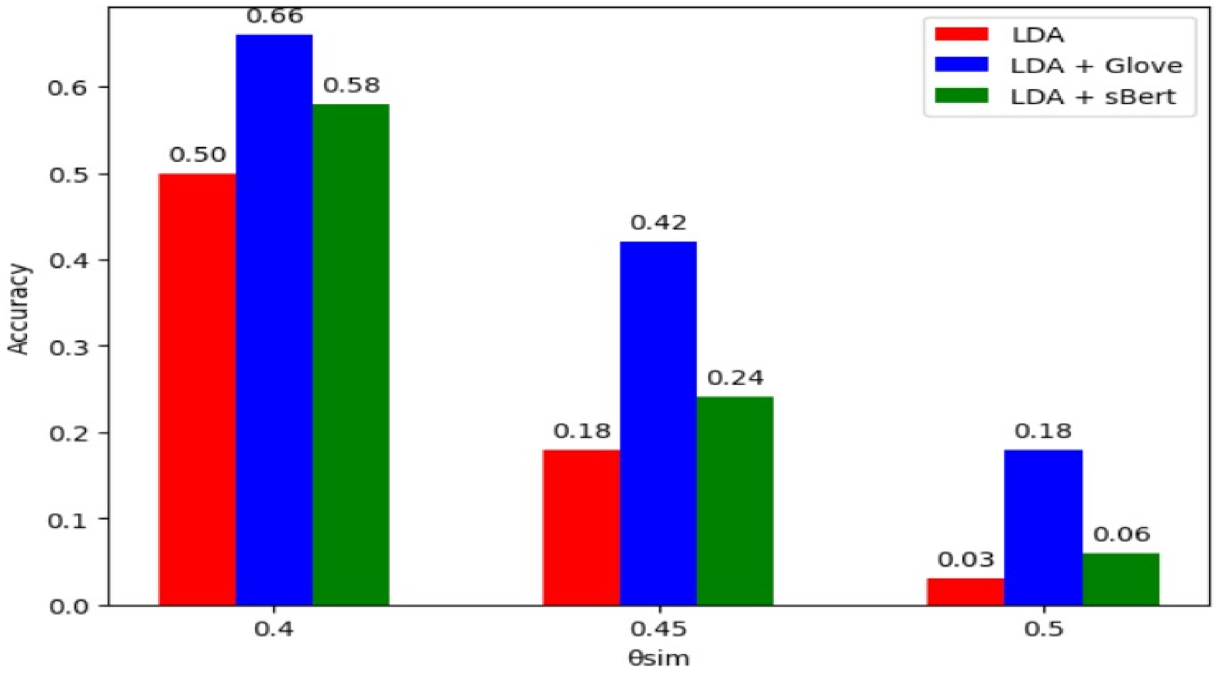


**Fig. 5.8** Accuracy of the Model using Algorithm II on $\theta_{alg}$=0.5 and $\theta_{sim} = \{0.4, 0.45, 0.5\}$ on manually labelled goldset

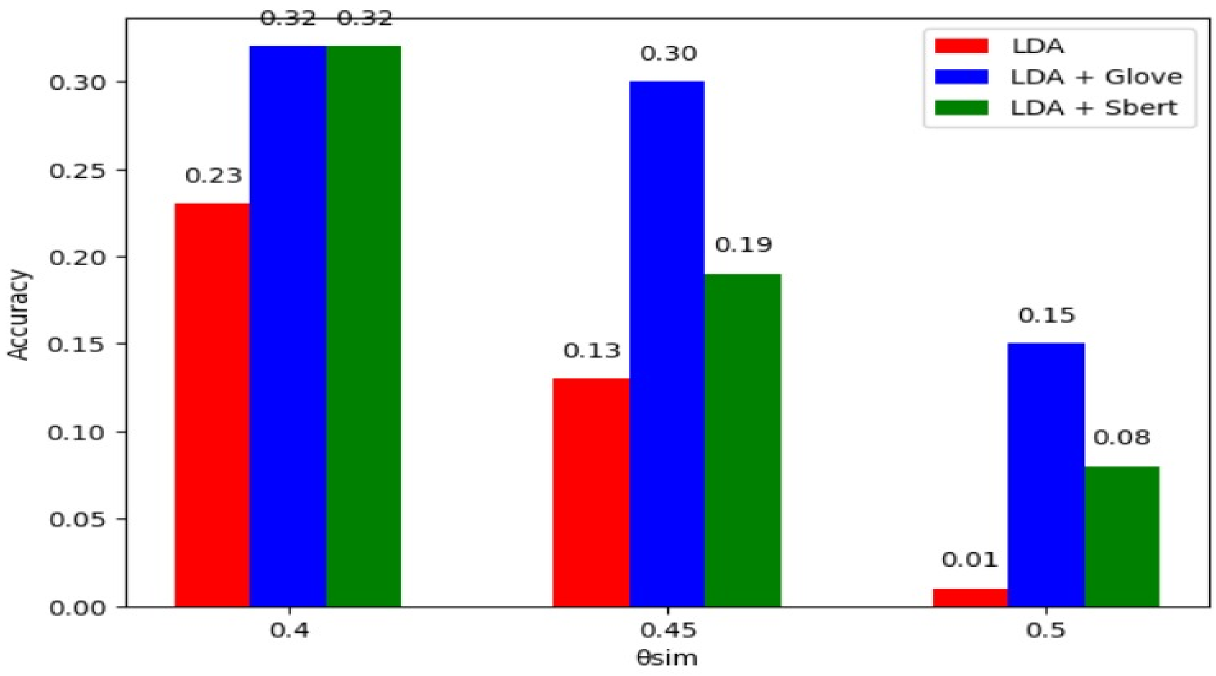


**Fig. 5.9** Accuracy of the model using Algorithm-III on $\theta_{alg}$=0.5 and $\theta_{sim} = \{0.4, 0.45, 0.5\}$

### 5.3.4 Statistics of GSDMM algorithm

Our method for identifying thematically similar generated sentences involved a two-stage process. In the first stage, we employed a technique to group the sentences into clusters based on their content. This clustering step ensures that sentences with similar themes are placed together. Following this initial clustering, we utilized two separate methods for tag prediction (thematic label assignment) within each cluster. The first method directly selected the four most probable tags from the cluster itself. The second method leveraged word embeddings (specifically, Glove) to identify the most relevant tags for each sentence. Figure 5.10 presents the resulting statistics on similarity accuracy achieved using both methods.

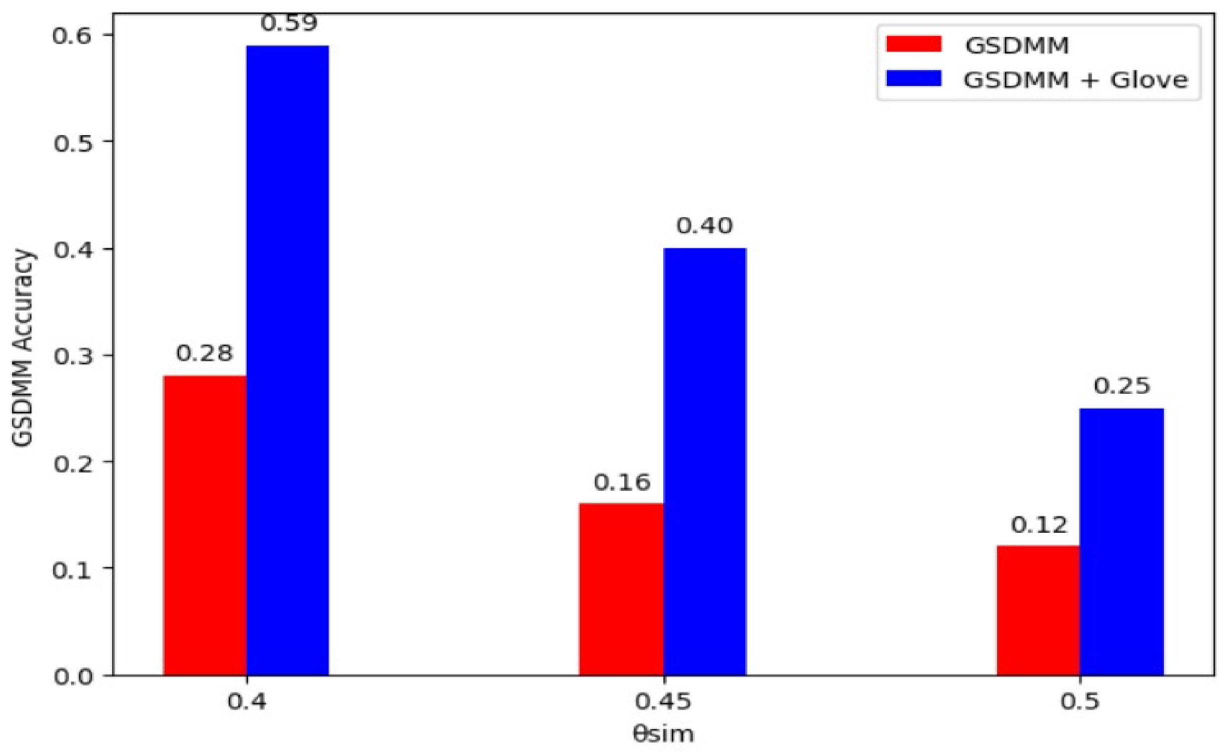


**Fig. 5.10** Accuracy of the GSDMM Model and $\theta_{sim} = \{0.4, 0.45, 0.5\}$

### 5.3.5 Comparison of Model Performance

Comparison of all algorithms with their best-performing model are shown in table 5.1 and Fig. 5.11.

**Table 5.1** Accuracy of Different Techniques with different $\theta_{alg}$ *and* $\theta_{sim}$

| Method | $\boldsymbol{\theta_{alg}}$ | $\boldsymbol{\theta_{sim}}$ | Accuracy |
|---|---|---|---|
| Algorithm-I + LDA + Glove | 0.0 | 0.4 | 0.41 |
| Algorithm-II + LDA | 0.5 | 0.4 | 0.76 |
| Algorithm-III + LDA + Glove | 0.5 | 0.4 | 0.32 |
| GSDMM | - | 0.4 | 0.28 |
| GSDMM + Glove | - | 0.4 | 0.59 |
| LDA | - | 0.4 | 0.27 |
| LDA + Glove | - | 0.4 | 0.36 |
| LDA + Sbert | - | 0.4 | 0.25 |

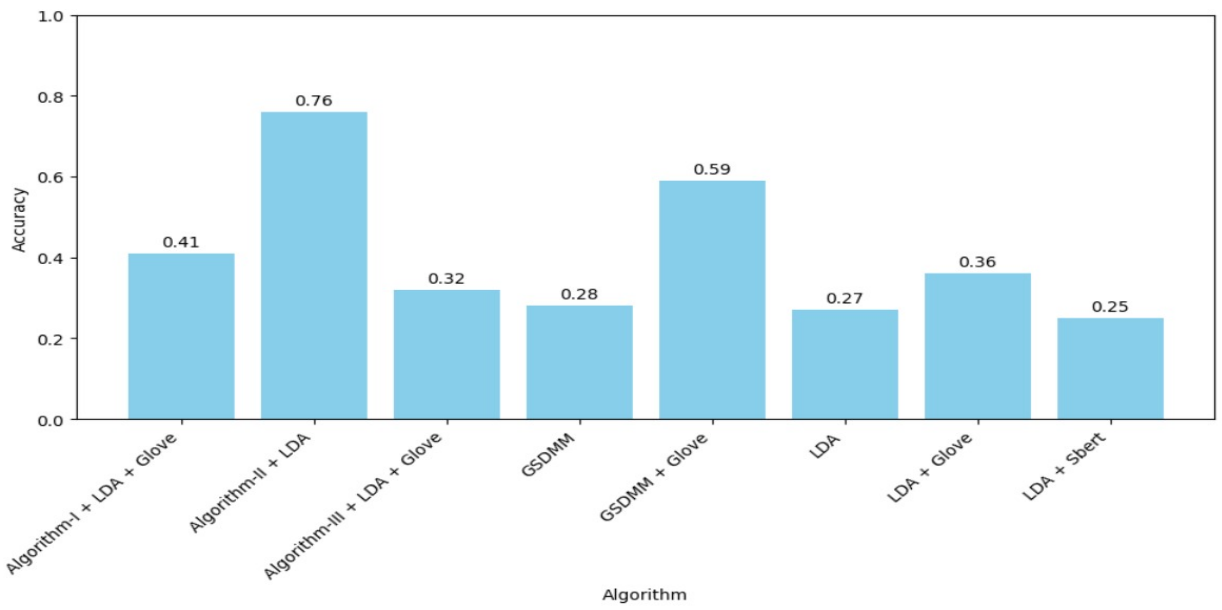


**Fig. 5.11** Overall Performance Comparison

# Chapter 6

# Conclusion and Future Work

Our initial exploration using the GSDMM model on the email dataset yielded suboptimal results in terms of topic labelling accuracy. To address this limitation, we investigated techniques for contextually enriching short text documents. We experimented with approaches leveraging LLaMA model with our proposed algorithms for short text extensions within the same semantic domain.

This research delves into the application of various algorithms and techniques to enhance the efficacy of document topic labelling. Future works are the following:

- Apply the LLaMA model and proposed algorithm for the email dataset.
- Building extension for the mail server.

# References

[1] “Lda model image,” https://en.wikipedia.org/wiki/File:Smoothed_LDA.png.

[2] J. Yin and J. Wang, “A dirichlet multinomial mixture model-based approach for short text clustering,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 233–242. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2623330.2623715

[3] D. Q. Nguyen, “jLDADMM: A Java package for the LDA and DMM topic models,” *arXiv preprint arXiv:1808.03835*, 2018.

[4] F. B. Valero, M. Baranes, and E. V. Epure, “Topic modeling on podcast short-text metadata,” in *Advances in Information Retrieval*, M. Hagen, S. Verberne, C. Macdonald, C. Seifert, K. Balog, K. Nørvåg, and V. Setty, Eds. Cham: Springer International Publishing, 2022, pp. 472–486. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-99736-6_32

[5] “Meta llama2 huggingface website,” https://huggingface.co/meta-llama/Llama-2-7b-chat-hf.

[6] “Google custom search official website,” https://developers.google.com/custom-search/v1.

[7] “gpt-3.5-turbo official website,” https://platform.openai.com/docs/models/gpt-3-5-turbo.

[8] “Tagmynews dataset,” https://github.com/AIRobotZhang/STCKA/blob/master/dataset/tagmynews.txt.