fLSA: Learning Semantic Structures in Document Collections Using Foundation Models

Weijia Xu, Nebojsa Jojic & Nicolas Le Roux
Microsoft Research
Redmond, WA 98052, USA
{weijiaxu,jojic,nleroux}@microsoft.com

Abstract

Humans can learn to solve new tasks by inducing high-level strategies from example solutions to similar problems and then adapting these strategies to solve unseen problems. Can we use large language models to induce such high-level structure from example documents or solutions? We introduce *fLSA*, a foundation-model-based Latent Semantic Analysis method that iteratively clusters and tags document segments based on document-level contexts. These tags can be used to model the latent structure of given documents and for hierarchical sampling of new texts. Our experiments on story writing, math, and multi-step reasoning datasets demonstrate that *fLSA* tags are more informative in reconstructing the original texts than existing tagging methods. Moreover, when used for hierarchical sampling, *fLSA* tags help expand the output space in the right directions that lead to correct solutions more often than direct sampling and hierarchical sampling with existing tagging methods.[1]

1 Introduction

Large language models (LLMs) have shown impressive performance on a wide range of tasks, such as reasoning (Suzgun et al., 2022; Liu et al., 2023), math problem solving (Wu et al., 2023), and open-ended text generation tasks (Katz et al., 2024; Dubey et al., 2024; OpenAI et al., 2024). Given natural language instructions or in-context examples with chain-of-thought steps, LLMs can adapt quickly to a new task and achieve outstanding performance on challenging tasks that require multi-step reasoning or planning (Wei et al., 2022). However, such methods typically rely on humans to induce the common strategy for solving a type of problems and demonstrate the strategy through few-shot chain-of-thought prompting. By contrast, humans learn to solve a new type of problems by analyzing some example problems and their solutions, inducing the common strategies (i.e. *latent semantic structure*) underlying these problem solutions, and testing them out on the new problems.

Inducing the latent semantic structure in a set of documents can be modeled as an unsupervised clustering and tagging problem, where given a set of coarsely segmented documents, we cluster the text segments that share common characteristics into the same set and assign a tag to each set of segments. Based on these segment tags, we can then uncover the latent structure by learning a dynamic model over the latent tags and their transition probabilities in the document set. As an example, Figure 1 shows a dynamic model over learned tags in mathematical solutions. Such dynamic models can help humans better understand and analyze large collections of documents. They also encode more generalizable information compared to few-shot examples, providing a useful guide for LLMs to solve new problems without manual intervention (as shown by the example in Figure 2). Additionally, they can also aid in searching algorithms on complex reasoning tasks (Guan et al., 2025) through hierarchical sampling: one can sample from the dynamic model over latent tags as an outline for the actual solution steps to explore more diverse solution paths during the rollout stage.

In this paper, we introduce *fLSA*, an iterative algorithm that alternatively clusters and tags document segments using LLMs based on segment- and document-level contexts. *fLSA* combines the merits of traditional topic modeling approaches such as Latent Semantic Analysis (LSA) (Hofmann et al., 1999) and LLM-based approaches, and captures shared semantic features among text segments more effectively. We evaluate 1) the informativeness of *fLSA* tags by measuring how well they help reconstruct the original text spans, and 2) their usefulness in expanding the search space in the right

[1]Code available at https://github.com/microsoft/fLSA.

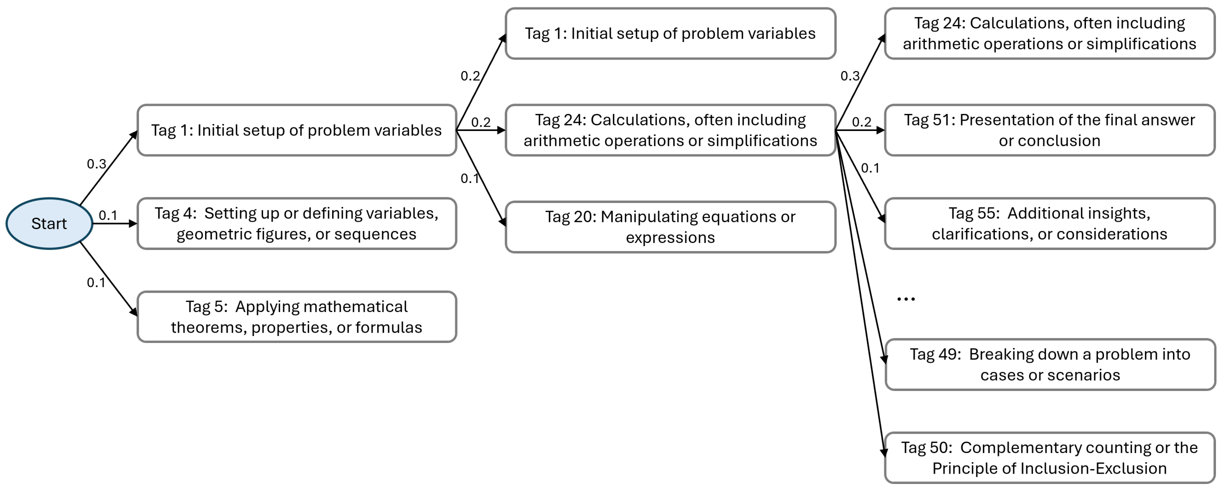
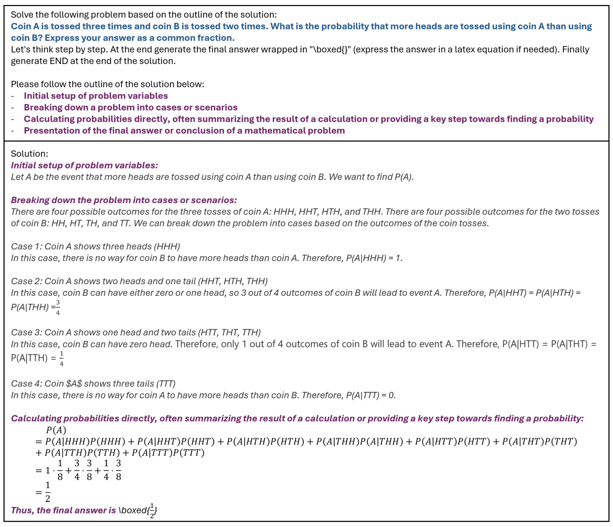

Start
0.3
0.1
0.1
Tag 1: Initial setup of problem variables
Tag 4: Setting up or defining variables, geometric figures, or sequences
Tag 5: Applying mathematical theorems, properties, or formulas
0.2
0.2
0.1
Tag 1: Initial setup of problem variables
Tag 24: Calculations, often including arithmetic operations or simplifications
Tag 20: Manipulating equations or expressions
0.3
0.2
0.1
Tag 24: Calculations, often including arithmetic operations or simplifications
Tag 51: Presentation of the final answer or conclusion
Tag 55: Additional insights, clarifications, or considerations
...
Tag 49: Breaking down a problem into cases or scenarios
Tag 50: Complementary counting or the Principle of Inclusion-Exclusion

Figure 1: Visualizing the bigram dynamic model over the latent tags learned on MATH solutions. For each tag, we list the three most probable next tags based on the transition probabilities $p(t_k|t_{k-1})$. The transition probabilities are annotated on the arrows. For Tag 24, we also list two example next tags outside the top-3 choices with transition probabilities $p \approx 0.01$.



Solve the following problem based on the outline of the solution:
Coin A is tossed three times and coin B is tossed two times. What is the probability that more heads are tossed using coin A than using coin B? Express your answer as a common fraction.
Let's think step by step. At the end generate the final answer wrapped in "\boxed{}" (express the answer in a latex equation if needed). Finally generate END at the end of the solution.

Please follow the outline of the solution below:
- **Initial setup of problem variables**
- **Breaking down a problem into cases or scenarios**
- **Calculating probabilities directly, often summarizing the result of a calculation or providing a key step towards finding a probability**
- **Presentation of the final answer or conclusion of a mathematical problem**

Solution:
Initial setup of problem variables:
Let A be the event that more heads are tossed using coin A than using coin B. We want to find P(A).

Breaking down the problem into cases or scenarios:
There are four possible outcomes for the three tosses of coin A: HHH, HHT, HTH, and THH. There are four possible outcomes for the two tosses of coin B: HH, HT, TH, and TT. We can break down the problem into cases based on the outcomes of the coin tosses.

Case 1: Coin A shows three heads (HHH)
In this case, there is no way for coin B to have more heads than coin A. Therefore, P(A|HHH) = 1.

Case 2: Coin A shows two heads and one tail (HHT, HTH, THH)
In this case, coin B can have either zero or one head, so 3 out of 4 outcomes of coin B will lead to event A. Therefore, P(A|HHT) = P(A|HTH) = P(A|THH) =$\frac{3}{4}$

Case 3: Coin A shows one head and two tails (HTT, THT, TTH)
In this case, coin B can have zero head. Therefore, only 1 out of 4 outcomes of coin B will lead to event A. Therefore, P(A|HTT) = P(A|THT) = P(A|TTH) = $\frac{1}{4}$

Case 4: Coin A shows three tails (TTT)
In this case, there is no way for coin A to have more heads than coin B. Therefore, P(A|TTT) = 0.

Calculating probabilities directly, often summarizing the result of a calculation or providing a key step towards finding a probability:

$$\begin{aligned}&P(A)\\&= P(A|HHH)P(HHH) + P(A|HHT)P(HHT) + P(A|HTH)P(HTH) + P(A|THH)P(A|THH) + P(A|HTT)P(HTT) + P(A|THT)P(THT)\\&+ P(A|TTH)P(TTH) + P(A|TTT)P(TTT)\\&= 1\cdot\frac{1}{8}+\frac{3}{4}\cdot\frac{3}{8}+\frac{1}{4}\cdot\frac{3}{8}\\&=\frac{1}{2}\end{aligned}$$

Thus, the final answer is *\boxed{$\frac{1}{2}$}*

Figure 2: An example of using the sampled tag sequence as an outline (in purple) to aid an LLM in generating a solution (italicized) to the given problem (in blue).

directions by measuring the Hits@K accuracy of the generated solutions through hierarchical sampling using the tags. Experiments on story writing, math and multi-step reasoning datasets show that *fLSA* leads to higher reconstruction likelihood than existing tagging approaches. Furthermore, on math and reasoning tasks, hierarchical sampling using *fLSA* tags helps expand the output space in the right directions more effectively than both direct sampling and existing tagging methods.

2 Related Work

2.1 Document Segmentation and Labeling

To model the structure and topic shifts in a document, prior work has introduced unsupervised document segmentation and labeling approaches that leverage term co-occurrence features (Hearst, 1997), co-occurrence shifts in topic vectors (Riedl and Biemann, 2012), lexical features and word embeddings (Glavaš et al., 2016). These approaches focus mostly on lexical features which are limited in modeling the high-level semantic structure of documents. On the other hand, Neural-based approaches have the potential of modeling sentence-level semantics and document-level topic flows more effective, but rely heavily on supervised training samples in the target domain (Koshorek et al., 2018; Arnold et al., 2019; Zhang et al., 2019). Our algorithm infers the structure of documents based on segment- and document-level contexts using LLMs in an unsupervised fashion.

2.2 Topic Modeling

Topic modeling is a widely used technique in natural language processing for uncovering hidden thematic structures in large text corpora. The most foundational methods in this domain include Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and Latent Semantic Analysis (LSA) (Hofmann et al., 1999; Hofmann, 1999, 2001). Both methods represent each document as a bag of words and models word-document relationships using a mixture of latent topics, where each topic is represented by a list of top words. These algorithms are mathematically grounded, but typically rely on manual topic interpretation, which often leads to incorrect or incomplete labels (Gillings and Hardie, 2022). More recent work introduces neural topic models (Miao et al., 2016; Dieng et al., 2020; Srivastava and Sutton, 2017), which combine traditional topic models with word embeddings. These models have shown improved performance in handling large and complex vocabularies. However, they still model each document as a bag of words, disregarding the sentence- and document-level semantics. Additionally, the resulting topics are represented either by semantic vectors or lists of closest words, which still rely on manual interpretation. Furthermore, studies have shown that incorporating expert knowledge in topic modeling improves over traditional unsupervised methods (Lee et al., 2017).

Moreover, the advent of large language models (LLMs) has led to LLM-based topic modeling approaches. Li et al. (2023) propose to use LLMs for topic labeling based their top terms produced by traditional topic models. For short text spans, however, the bag-of-words representation of texts provides limited information for topic modeling. Akash et al. (2023) address the issue by extending each text span into longer sequences using LLMs and extracting topics from the extended texts using neural topic models. Futhermore, Pham et al. (2024); Wang et al. (2023); Mu et al. (2024) propose prompt-based techniques to generate, merge, and assign topics using LLMs. These approaches leverage the domain knowledge embedded in LLMs and produce more interpretable topics based on sentence or document-level contexts beyond bag of words.

However, the generate-and-merge approach limits the model's potential for discovering shared features among various text spans across documents of different themes and often leads to overly abstract, thematical topics, especially on a large-scale document collection. We propose *fLSA*, which combines the merits of traditional LSA, which uses an iterative EM algorithm to model topic and text distributions, and LLM-based approaches.

3 Approach

We propose *fLSA*, a foundation-model-based EM algorithm that learns the latent tags on a set of segmented documents. We draw inspiration from the traditional Probabilistic Latent Semantic Analysis and use iterative EM steps to learn the latent tags that maximize the estimated likelihood of segmented documents.

3.1 Probabilistic Latent Semantic Analysis (PLSA)

PLSA models the distribution over words w in a document d as a mixture of conditionally independent multinomial distributions, each such distribution representing a *topic* t. This generative model of words in a document is usually expressed mathematically in terms of the distribution:

$$p_\Theta(w|d) = \sum_t p_\Theta(t|d) p_\Theta(w|t), \tag{1}$$

which can be sampled by first sampling a topic t for the given document d from $p_\Theta(t|d)$ and then sampling words conditioned on the topic from $p_\Theta(w|t)$. Θ represents the parameters of the PLSA model. PLSA aims to find Θ that maximizes the log-likelihood of words in all documents:

$$\mathcal{L} = \sum_{d,w} \log \sum_t p_\Theta(t|d) p_\Theta(w|t) \tag{2}$$

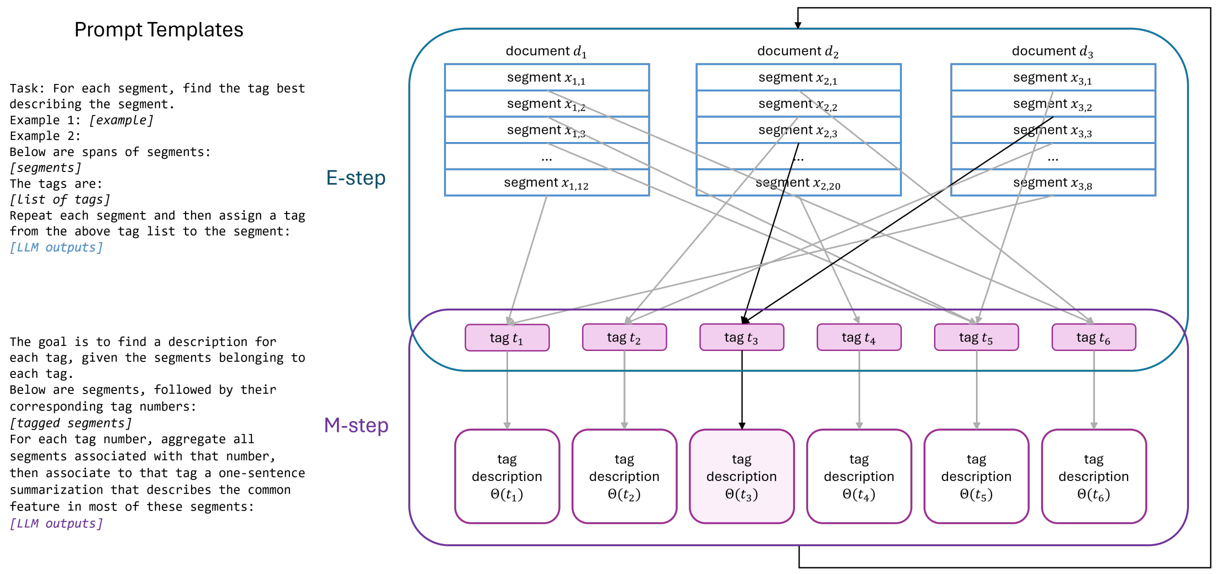
To estimate the parametric distributions $p_\Theta(t|d)$ and $p_\Theta(w|t)$, PLSA relies on an EM algorithm, which is an iterative method to find the maximum likelihood estimate of parameters in statistical models. Specifically, an EM iteration alternates between an expectation (E) step and a maximization (M) step. At iteration i, the E-step estimates the posterior distribution $p_{\Theta_{i-1}}(t|w,d)$ of topics t conditioned on each document d and word w in it based on fixed parameters Θ_{i-1} from the previous iteration:

$$p_{\Theta_{i-1}}(t|w,d) = \frac{p_{\Theta_{i-1}}(t|d) p_{\Theta_{i-1}}(w|t)}{\sum_{t'} p_{\Theta_{i-1}}(t'|d) p_{\Theta_{i-1}}(w|t')} \tag{3}$$

The M-step optimizes the parameters Θ such that the expectation of the log-likelihood $p_\Theta(w|d)$ of words in each document given t sampled from the estimated posterior $p_{\Theta_{i-1}}(t|w,d)$ is maximized:

$$\arg\max_\Theta \sum_{d,w} \mathbb{E}_{t\sim p_{\Theta_{i-1}}(t|w,d)} \log p_\Theta(t|d) p_\Theta(w|t) \tag{4}$$

Theoretically, each EM iteration will yield a larger likelihood in Eq 2 until it converges to a local maximum. In topic modeling literature, various generalized EM variants exist, including the ones that approximate the posterior distribution with a small number of samples, or just the mode of it, and which alter the parameters so that they do not necessarily maximize the likelihood under the posterior, but simply improve it.


Prompt Templates

```
Task: For each segment, find the tag best
describing the segment.
Example 1: [example]
Example 2:
Below are spans of segments:
[segments]
The tags are:
[list of tags]
Repeat each segment and then assign a tag
from the above tag list to the segment:
[LLM outputs]
```

```
The goal is to find a description for
each tag, given the segments belonging to
each tag.
Below are segments, followed by their
corresponding tag numbers:
[tagged segments]
For each tag number, aggregate all
segments associated with that number,
then associate to that tag a one-sentence
summarization that describes the common
feature in most of these segments:
[LLM outputs]
```

Figure 3: An illustration of the E-step and M-step in *fLSA*. At the E-step, we assign each text segment to a tag through prompting given the tag descriptions at the previous iteration. At the M-step, we prompt the LLM to generate new tag descriptions based on the segments assigned to each tag at the E-step.

3.2 Foundation-Model-Based LSA (fLSA)

We introduce *fLSA*, which learns the latent *tags* (similar to *topics* in LSA)[2] on a set of segmented documents $d = (x_1, x_2, ..., x_L)$, where the document d is segmented into L segments x_k. A core difference between *fLSA* and PLSA is that PLSA models the generative probability of each word in a document independently, while *fLSA* models the probability of the sequence of words $(w_1, w_2, ..., w_n)$ in each text segment x_k jointly as $p_\Theta(w_1, w_2, ..., w_n|t)$. Moreover, PLSA models the distribution over tags $p_\Theta(t|d)$ for each document independently of other documents, while *fLSA* models the distribution over tags t conditioned not only on current segment x_k but also on the document d.

To express the difference mathematically, in *fLSA*, the generative model of a segment $x_k = w_{1..n}$ in a document d can be written as:

$$p_\Theta(w_{1..n}|x_k, d) = \sum_t p_\Theta(t|x_k, d)p_\Theta(w_{1..n}|t), \tag{5}$$

which can be sampled by first sampling a tag t for the current segment x_k in document d and then sampling the word sequence $w_{1..n}$ for that segment given the tag.

Another core difference between *fLSA* and PLSA is that we model the parametric distributions $p_\Theta(t|x_k, d)$ and $p_\Theta(w_{1..n}|t)$ using an LLM with frozen parameters, and the tunable "parameters" Θ in *fLSA* are the *textual* description $\Theta(t)$ for each tag t and the tag assignment for each segment.

Analogously to the (generalized) EM algorithms for traditional topic models, we are seeking Θ that corresponds to high likelihood of the word sequence in each document:

$$\mathcal{L} = \sum_{d,x_k} \log \sum_t p_\Theta(t|x_k, d)p_\Theta(w_{1..n}|t) \tag{6}$$

Our iterative EM steps are shown in Figure 3. At the E-step in iteration i, we approximate the posterior distribution $p_{\Theta_{i-1}}(t|w_{1..n}, x_k, d)$ of tags t for each segment $x_k = w_{1..n}$ in document d by prompting the LLM to greedily assign a tag given the tag descriptions $\Theta_{i-1}(t)$ from the previous iteration, the current segment $x_k = w_{1..n}$ and neighbouring segments $(x_{k-W/2}, x_{k+1-W/2}, ..., x_{k+W/2})$ as document-level context, where W is the context window size.[3] At the M-step, in lieu of maximizing (or just improving) the expected log-likelihood $p_\Theta(w_{1..n}|x_k, d)$ of words in each segment given the tag assignments from the E-step,

$$\arg\max_\Theta \sum_{d,x_k} \mathbb{E}_{t\sim p_{\Theta_{i-1}}(t|w_{1..n},x_k,d)} \log p_\Theta(t|x_k, d)p_\Theta(w_{1..n}|t), \tag{7}$$

we obtain updated tag descriptions $\Theta(t)$ by inviting the LLM itself to summarize the segments assigned

[2]We use the terminology *tag* instead of *topic* in our algorithm because they may cover shared characteristics among document segments beyond topics (see the example tags in Figure 1).

[3]At the first iteration, since the tag descriptions are empty, we assign tags randomly.

to the tag t: We aggregate the segments assigned to tag t and prompt the LLM to generate a tag description that best summarizes what these segments share in common (Fig. 3).

4 Experimental Setup

4.1 Datasets

We evaluate *fLSA* against various baselines on WritingPrompts (a story writing dataset (Fan et al., 2018)), MATH (which contains math problems and the corresponding solution texts (Hendrycks et al., 2021)), and Big-Bench Hard (BBH) benchmark (which contains diverse types of reasoning problems and their solutions (Suzgun et al., 2022)). We set the number of tags to 100 for WritingPrompts and MATH, and 50 for BBH (see the Appendix for more details).

4.2 Evaluation Metrics

Reconstruction Likelihood To measure the informativeness of learned tags (either through *fLSA* or a baseline algorithm), we measure the reconstruction log-likelihood of the test documents (stories in the test set of WritingPrompts or problem solutions in the test set of MATH) conditioned on the tags.

Specifically, for each test case x_k, which is a segment randomly sampled from a test document $x_{1...L}$ (randomly sampled from the test corpus), we approximate the reconstruction log-likelihood of x_k given latent tags t_k predicted given x_k and its neighboring segments under the LLM:

$$\mathbb{E}_{t_k \sim p_{LLM}(t|x_k,d)}[\log p_{LLM}(x_k|x_{1...k-1}, t_k)] \quad (8)$$

Specifically, we first sample S alternative segments at position k independently by $\{\tilde{x}_k^{(1)}, \tilde{x}_k^{(2)}, ..., \tilde{x}_k^{(S)}\} \sim p_{LLM}(\cdot|x_{1...k-1})$. Next, we conduct T repeated experiments to approximate the log-likelihood of x_k given the previous segments $x_{1...k-1}$ and the tag t_k predicted on x_k under the LLM. Each time, we randomly sample C alternative segments from $\{\tilde{x}_k^{(1)}, \tilde{x}_k^{(2)}, ..., \tilde{x}_k^{(S)}\}$ and put it together with x_k (in randomly shuffled order) as options and ask the LLM which one is the true continuation conditioned on $x_{1...k-1}$ and t_k. Based on the number of times (denoted as c_k) that the LLM chooses x_k as the true continuation among all T experiments, we estimate the reconstruction log-likelihood with alpha-smoothing ($\alpha = 0.1$):

$$\begin{aligned}&\mathbb{E}_{t_k \sim p_{LLM}(t|x_k,d)}[\log p_{LLM}(x_k|x_{1...k-1}, t_k)]\\ &= \log \frac{c_k + \alpha}{T + \alpha S}\end{aligned} \quad (9)$$

As a baseline, we compare the reconstruction log-likelihood with the log-likelihood computed the same way as above but without conditioning on any tags:

$$\mathbb{E}[\log p_{LLM}(x_k|x_{1...k-1})] = \log \frac{c'_k + \alpha}{T + \alpha S} \quad (10)$$

where c'_k is the number of times that the LLM chooses x_k as the true continuation among T experiments, which is computed the same way as above except that when asking the LLM to choose the true continuation, we only provide the previous text segments $x_{1...k-1}$ without any tags.

In our experiments, we evaluate the reconstruction log-likelihood of all methods on the same set of 1K randomly sampled test cases.

Hits@K Accuracy To demonstrate that the learned tags can also help expand the search space in the right directions when searching for effective solutions to a complex reasoning task, we learn a dynamic model over the latent tags (as shown by the example in Figure 1) and use it for hierarchical sampling, where we first sample a sequence of tags as an outline and then sample the actual text based on the outline. And then, we evaluate the Hits@K accuracy of hierarchical sampling with latent tags, and compare it with the Hits@K accuracy of direct sampling without tags. Specifically, for each problem, we sample $K = 50$ solutions independently from an LLM given the problem description either directly or through hierarchical sampling with latent tags. If any of the K solutions leads to the correct answer, it gets a score of 1, otherwise 0. Finally, we compute the average score over all testing problems.

For hierarchical sampling, we first sample a sequence of tags $(t_1, t_2, ..., t_l)$ (up till the special tag <END>) with maximum length L using a bigram model learned on the training data (without conditioning on the test problem):

$$\begin{aligned}&p(t_1, t_2, ..., t_l)\\ =&p(t_1)p(t_2|t_1)...p(t_l|t_{l-1})p(\text{<END>}|t_l)\end{aligned} \quad (11)$$

And then, we prompt the LLM to generate a solution to the given problem based on the tag sequence $(t_1, t_2, ..., t_l)$ using the prompt template shown in Figure 2.

4.3 *fLSA* Setup

For the EM procedure, we set the maximum number of iterations to 30.[4] At the E-step (where the LLM assigns a tag to each segment conditioned not only on the current segment but also on neighbouring segments within the context window), we use a context window size of 2 on WritingPrompts and use unlimited context window (such that the whole solution is used as context) on MATH and BBH. At the M-step, we randomly sample 10 segments assigned to each tag to update the tag description.

4.4 Baselines

TradLDA We compare our approach with the traditional Latent Dirichlet Allocation (*TradLDA*), a type of LSA algorithm designed to discover latent topics in a collection of text spans (Blei et al., 2003).

TradLDA+LLM As Li et al. (2023) showed that the topic labels generated by LLMs based on the key terms learned through TradLDA are preferred more often than the original labels, we also include *TradLDA+LLM* as a baseline. Specifically, we first learn the topics and the key terms for each topic using TradLDA, and then use GPT-4 to generate a description for each topic based on the key terms.

Prompting Recent work showed that, with appropriate prompts, LLMs are capable of directly generating topic labels given a set of text documents and condensing overarching topics (Pham et al., 2024; Wang et al., 2023; Mu et al., 2024). As a baseline, we adapt the approach (along with the prompts) in Mu et al. (2024) to generate topic descriptions for each text segment.

GenOutline For Hits@K accuracy, we also include a two-step sampling baseline, where we first prompt the LLM to generate a multi-step outline for solving this type of problem and then prompt the LLM to generate the actual solution based on the problem description and the outline.

4.5 Large Language Model Setup

For clustering and tagging, we use GPT-4 (OpenAI et al., 2024) and Qwen-2.5-7B (a much smaller LLM introduced in Qwen et al. (2025)). We also use GPT-4 to estimate the reconstruction log-likelihood. To measure Hits@K Accuracy, we use ChatGPT (gpt-3.5-turbo; OpenAI (2023)) instead of GPT-4, because GPT-4 has achieved high accuracy on MATH and BBH (e.g. 84% on MATH (Zhou et al., 2023)), possibly due to data contamination issues (Deng et al., 2024; Bubeck et al., 2023). Thus, we use ChatGPT for solution sampling to show the potential of using learned tags to diversify the sampled outputs and improve the chance of finding a correct answer when the model cannot find it through direct sampling.[5]

5 Results

5.1 Reconstruction Likelihood

First, we compare the reconstruction log-likelihood of *fLSA* with the *No Tag* baseline (without conditioning on any tags). As shown in Table 1, conditioning on *fLSA* tags helps predict the original texts: *fLSA* brings 0.7–1.4 higher log-likelihood than the *No Tag* baseline.

TradLDA also brings higher reconstruction log-likelihood over the *No Tag* baseline. However, since *TradLDA* only captures word or term co-occurrences, it still underperforms *fLSA* consistently on all three datasets. Moreover, *TradLDA+LLM* fails to improve over *TradLDA*. As shown by the examples in Table 2, it is extremely challenging for LLMs and even humans to extract meaningful semantic information from the key terms learned on short text segments through *TradLDA*, and the resulting tag descriptions are overly generic, making it challenging to reconstruct the original text segments accurately.

Compared with the Prompting baseline, *fLSA* achieves 0.2–0.5 higher log-likelihood on all three datasets. We further compared the tags learned using Prompting versus *fLSA*. As shown by the examples in Table 3, Prompting tends to merge unrelated topics into a mixed topic (e.g. Tag 1 and 2), and the resulting topics become overly broad. Even for tags sharing a common theme, the descriptions often lack specificity and detail (e.g. Tag 3). By contrast, *fLSA* identifies segments with similar themes, groups them into a single cluster and produces more detailed tag descriptions with example plots.

5.2 Hits@K Accuracy

We further evaluate how the tags and semantic structure learned through *fLSA* help expand the output space in the right directions that lead to

[4]We found in our preliminary experiment that the learned tag descriptions become stable (with very little semantic changes) in less than 30 iterations.

[5]More details in the Appendix.

	No Tag	TradLDA	TradLDA+LLM	Prompting	*fLSA*
WritingPrompts	-4.81	-3.75	-4.12	-3.62	**-3.43**
MATH-Num	-3.32	-2.96	-3.28	-3.06	**-2.64**
MATH-All	-3.67	-3.16	-3.57	-3.44	**-2.94**

Table 1: Reconstruction log-likelihood of *fLSA* versus the baseline without tags (*No Tag*), traditional LDA (*TradLDA*), traditional LDA with LLM-generated tag descriptions (*TradLDA+LLM*) (Li et al., 2023), and the prompting baseline (*Prompting*) (Mu et al., 2024) on *WritingPrompts* story dataset, Number Theory dataset from MATH (*MATH-Num*), and the MATH (*MATH-All*) dataset.

Key Terms	**Tag Description**
nothing, get, life, else, light, across, best, ca, single, come, got, death, together, running, power, system, entire, could, control, everything	The words you've provided span a broad range of concepts, but they share a common denominator in that they can all be associated with themes commonly found in science fiction literature and media.
continued, surface, wait, raised, floor, slowly, give, new, sure, needed, around, also, face, body, fact, made, bitch, girl, guy, much	The words listed seem to be common English words that could appear in a wide range of contexts. However, given their generic nature, they could be particularly prevalent in narrative or descriptive writing, such as in fiction, storytelling, or personal narratives.

Table 2: Examples of key terms learned on short story segments in WritingPrompts through *TradLDA* and the corresponding tag descriptions generated by GPT-4. Given only the key terms without context, the tag descriptions produced by GPT-4 are too generic to recover the original text spans.

Prompting Tags	***fLSA* Tags**
Tag 1: Stories involving themes of sacrifice, duty, friendship, companionship, hope, and resilience in the face of crisis.	Tag 1: Scenes involving intense, often dangerous situations, like explosions, retreats, long nights, empty streets, fires, and storms.
Tag 2: Stories involving time travel, genetic irregularities, and strange creatures that feed on negative emotions.	Tag 2: The protagonist experiences surreal and unexpected events, often involving time travel or strange bodily functions, and narrates them in a casual, humorous tone.
Tag 3: Stories involving emotional moments and first hugs.	Tag 3: This tag is associated with story segments that feature intense emotional moments, often involving fear, anger, or distress, and frequently serve as turning points or climactic scenes in the narrative.

Table 3: Example tags learned on short story segments in WritingPrompts through Prompting versus *fLSA*. Prompting tags are either too mixed (e.g. Tag 1 and 2) or too generic (e.g. Tag 3), while *fLSA* groups segments of similar themes into the same cluster and describes each cluster with detailed explanations and example plots.

correct solutions by measuring the Hits@K Accuracy of various sampling methods with or without tags. First, compared with direct sampling without using any tags, hierarchical sampling with *fLSA* tags leads to significantly higher Hits@K accuracy by +10.0 points on MATH and +16.6 points on BBH on average. Additionally, we compare *fLSA* with GenOutline, a two-step sampling approach where we prompt the LLM to generate an outline before generating the actual solution. GenOutline improves over direct sampling on most tasks, but still underperforms hierarchical sampling with

	No Tag	GenOutline	TradLDA	TradLDA+LLM	Prompting	*fLSA*
MATH						
Algebra	88.6	90.1	**93.6**	89.6	91.1	90.1
Counting	61.3	60.4	69.8	65.1	69.8	**70.8**
Geometry	53.1	55.2	58.3	57.3	**62.5**	60.4
InterAlgebra	55.7	51.7	58.7	59.2	**61.2**	**61.2**
Number	65.4	76.0	77.9	74.0	78.8	**83.7**
PreAlgebra	74.2	79.1	81.3	81.3	84.6	**89.0**
PreCalculus	42.2	46.8	51.4	46.8	49.5	**55.0**
Average	62.9	65.6	70.1	67.6	71.1	**72.9**
BBH						
Date	92.8	94.4	95.6	95.2	95.2	**98.8**
Formal	45.2	61.2	65.6	52.8	57.2	**93.2**
Geometric	70.8	76.8	83.6	84.0	80.0	**87.6**
Logical	89.2	95.6	95.6	96.0	96.5	**99.5**
Movie	84.8	88.0	92.8	92.0	93.2	**95.2**
ObjCount	93.2	96.8	99.2	**100.0**	**100.0**	95.2
Penguins	93.8	99.3	99.3	**100.0**	99.3	99.3
ReasonColored	92.8	97.6	98.4	98.8	98.8	**100.0**
RuinNames	64.8	74.8	69.6	70.0	80.0	**93.6**
TranslationError	52.4	68.4	60.4	60.0	63.6	**75.2**
Temporal	86.4	98.4	93.2	96.8	98.0	**100.0**
WordSort	27.2	36.4	16.0	14.8	42.0	**56.0**
Average	74.5	82.3	80.8	80.0	83.7	**91.1**

Table 4: Hits@K accuracy of *fLSA* versus directly sampling without tags (*No Tag*), two-step sampling with LLM-generated outline (*GenOutline*), traditional LDA (*TradLDA*), traditional LDA with LLM-generated tag descriptions (*TradLDA+LLM*) (Li et al., 2023), and the prompting baseline (*Prompting*) (Mu et al., 2024) on 12 challenging tasks from BBH benchmark (Suzgun et al., 2022) and 7 tasks from MATH (Hendrycks et al., 2021).

fLSA by 7–9 points. These results indicate that hierarchical sampling using tags derived from the domain-specific documents via *fLSA* produces more effective output solutions, thereby increasing the likelihood of hitting the correct answer with K samples.

Next, we compare *fLSA* with hierarchical sampling with existing tagging approaches. *fLSA* tags expand the output space in the directions that lead to correct answers more often than TradLDA on 16 out of 19 tasks. It brings a significant improvement of 3–10 points over TradLDA.[6] Similarly, compared with TradLDA+LLM, *fLSA* achieves higher Hits@K Accuracy on 17 out of 19 tasks and improves the average accuracy by 5–11 points across BBH and MATH. Compared with the Prompting baseline, *fLSA* achieves higher Hits@K Accuracy on 14 out of 19 tasks. Overall, hierarchical sampling with *fLSA* tags improves Hits@K Accuracy significantly over existing tagging approaches by 2–11 points on average.

5.3 Learning Tags with Smaller LLMs

In addition to GPT-4, we also evaluate *fLSA* using a smaller LLM – Qwen-2.5-7B. We run *fLSA* using Qwen-2.5-7B as the base model (while the other hyper-parameters remain unchanged) and measure the Hits@K Accuracy of hierarchical sampling using the learned tags on BBH. We discover that the average accuracy drops by 5 points compared to the tags learned using GPT-4, but it still outperforms TradLDA and Prompting (using the much larger GPT-4 model) by 3–6 points.

5.4 Ablation Study

We further examine how the number of tags learned through *fLSA* influences its ability to expand the

[6] We test significance using paired student's t-test with significance level $\alpha = 0.05$.

output space. Specifically, we compare the Hits@K Accuracy of hierarchical sampling with 20, 50, and 100 *fLSA* tags on BBH tasks. Results show that the accuracy drops by 3 points when using 20 instead of 50 tags, whereas increasing the number of tags from 50 to 100 yields minimal change (see the Appendix for detailed results). This suggests that learning a sufficient – even redundant – number of tags can be beneficial for effectively expanding the output space.

6 Conclusion

We introduced *fLSA*, a foundation-model-based Latent Semantic Analysis method that aims to uncover the latent semantic structures in document collections by iteratively clustering and tagging document segments based on document-level contexts. Our experiments on story writing, math and multi-step reasoning tasks show that *fLSA* tags are more informative in reconstructing the original texts than tags generated by existing tagging methods. *fLSA* tags are also useful in expanding the output space via hierarchical sampling to increase the likelihood of discovering correct solutions to complex reasoning problems. These results suggest the potential of *fLSA* for generating effective task guidelines given some worked-out examples, along with hierarchical sampling and searching for problem solutions on challenging reasoning tasks.

7 Limitations

One limitation of *fLSA* is that some of the tags produced by *fLSA* may be semantically similar to each other, which can be ideally merged into a single tag. This limitation could be addressed by incorporating a tag fusion step in the EM algorithm, which we leave for future work. In addition, although the *fLSA* algorithm is agnostic to the LLM being used, we only test it on GPT-4 (which is one of the most powerful and widely used LLMs). Testing the algorithm on smaller models can be an interesting future work.

This work also has potential risks. One major risk is that the tags learned using *fLSA* may reflect the undesirable biases within the LLM being used. Integrating bias detection and mitigation techniques within the algorithm could be useful for addressing the issue.

References

Pritom Saha Akash, Jie Huang, and Kevin Chen-Chuan Chang. 2023. Let the pretrained language models "imagine" for short texts topic modeling. *Preprint*, arXiv:2310.15420.

Sebastian Arnold, Rudolf Schneider, Philippe Cudré-Mauroux, Felix A. Gers, and Alexander Löser. 2019. SECTOR: A neural model for coherent topic segmentation and classification. *Transactions of the Association for Computational Linguistics*, 7:169–184.

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

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *Preprint*, arXiv:2303.12712.

Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. 2024. Investigating data contamination in modern benchmarks for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8706–8719, Mexico City, Mexico. Association for Computational Linguistics.

Adji B. Dieng, Francisco J. R. Ruiz, and David M. Blei. 2020. Topic Modeling in Embedding Spaces. *Transactions of the Association for Computational Linguistics*, 8:439–453.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Mathew Gillings and Andrew Hardie. 2022. The interpretation of topic models for scholarly analysis: An evaluation and critique of current practice. *Digital Scholarship in the Humanities*, 38(2):530–543.

Goran Glavaš, Federico Nanni, and Simone Paolo Ponzetto. 2016. Unsupervised text segmentation using semantic relatedness graphs. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 125–130, Berlin, Germany. Association for Computational Linguistics.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *Preprint*, arXiv:2501.04519.

Marti A. Hearst. 1997. Text tiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. *CoRR*, abs/2103.03874.

T Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*.

Thomas Hofmann. 2001. Unsupervised learning by probabilistic latent semantic analysis. *Machine learning*, 42:177–196.

Thomas Hofmann et al. 1999. Probabilistic latent semantic analysis. In *UAI*, volume 99, pages 289–296.

Daniel Martin Katz, Michael James Bommarito, Shang Gao, and Pablo Arredondo. 2024. Gpt-4 passes the bar exam. *Philosophical Transactions of the Royal Society A*, 382(2270):20230254.

Omri Koshorek, Adir Cohen, Noam Mor, Michael Rotman, and Jonathan Berant. 2018. Text segmentation as a supervised learning task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 469–473, New Orleans, Louisiana. Association for Computational Linguistics.

Tak Yeon Lee, Alison Smith, Kevin Seppi, Niklas Elmqvist, Jordan Boyd-Graber, and Leah Findlater. 2017. The human touch: How non-expert users perceive, interpret, and fix topic models. *International Journal of Human-Computer Studies*, 105:28–42.

Dai Li, Bolun Zhang, and Yimang Zhou. 2023. Can large language models (llm) label topics from a topic model?

Hanmeng Liu, Ruoxi Ning, Zhiyang Teng, Jian Liu, Qiji Zhou, and Yue Zhang. 2023. Evaluating the logical reasoning ability of chatgpt and gpt-4. *arXiv preprint arXiv:2304.03439*.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1727–1736, New York, New York, USA. PMLR.

Yida Mu, Chun Dong, Kalina Bontcheva, and Xingyi Song. 2024. Large language models offer an alternative to the traditional approach of topic modelling. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 10160–10171, Torino, Italia. ELRA and ICCL.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Chau Pham, Alexander Hoyle, Simeng Sun, Philip Resnik, and Mohit Iyyer. 2024. TopicGPT: A prompt-based topic modeling framework. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2956–2984, Mexico City, Mexico. Association for Computational Linguistics.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Martin Riedl and Chris Biemann. 2012. TopicTiling: A text segmentation algorithm based on LDA. In *Proceedings of ACL 2012 Student Research Workshop*, pages 37–42, Jeju Island, Korea. Association for Computational Linguistics.

Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. *Preprint*, arXiv:1703.01488.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Han Wang, Nirmalendu Prakash, Nguyen Khoi Hoang, Ming Shan Hee, Usman Naseem, and Roy Ka-Wei Lee. 2023. Prompting large language models for topic modeling. In *2023 IEEE International Conference on Big Data (BigData)*, pages 1236–1241.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Yiran Wu, Feiran Jia, Shaokun Zhang, Hangyu Li, Erkang Zhu, Yue Wang, Yin Tat Lee, Richard Peng, Qingyun Wu, and Chi Wang. 2023. An empirical study on challenging math problem solving with gpt-4. *arXiv preprint arXiv:2306.01337*.

Weijia Xu, Andrzej Banburski, and Nebojsa Jojic. 2024. Reprompting: Automated chain-of-thought prompt inference through Gibbs sampling. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 54852–54865. PMLR.

Ruqing Zhang, Jiafeng Guo, Yixing Fan, Yanyan Lan, and Xueqi Cheng. 2019. Outline generation: Understanding the inherent content structure of documents. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 745–754.

Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, and Hongsheng Li. 2023. Solving challenging math word problems using gpt-4 code interpreter with code-based self-verification. *Preprint*, arXiv:2308.07921.

A Appendix

A.1 Datasets

We evaluate *fLSA* against various baselines on story writing, math problem solving and multi-step reasoning benchmarks. We use WritingPrompts (Fan et al., 2018), a story writing dataset that contains 300K human-written stories paired with writing prompts from an online forum. We randomly sample 100 stories from the training set for clustering and tagging. We set the number of tags to 100 for all tagging approaches. For math problem solving, we use MATH (Hendrycks et al., 2021), a popular math benchmark that contains high school math competition problems on seven subjects including Prealgebra, Algebra, Number Theory, Counting and Probability, Geometry, Intermediate Algebra and Precalculus. We learn 100 tags on 1K randomly sampled problem solutions from the training set. We also experiment on the Big-Bench Hard (BBH) benchmark (Suzgun et al., 2022). The original benchmark includes 23 challenging multi-step reasoning tasks, but each task only includes three step-by-step solution examples. Instead, we take the 12 tasks used in Xu et al. (2024) and learn the tags on the problem solutions (produced by their automatic prompt inference algorithm) for the 179 training problems. We set the number of tags to 50 for BBH.[7]

A.2 Large Language Model Setup

For clustering and tagging, we use GPT-4 (OpenAI et al., 2024) and Qwen-2.5-7B (a much smaller LLM introduced in Qwen et al. (2025)). For GPT-4, we set $top\_p = 0.5$, sampling temperature $\tau = 1.0$, zero frequency and presence penalty. For Qwen-2.5-7B, we set $top\_p = 0.5$, sampling temperature $\tau = 0.1$, zero frequency and presence penalty.

We also use GPT-4 with $top\_p = 0.5$ to estimate the reconstruction log-likelihood. We set the temperature $\tau = 1.0$ when sampling alternative segments and $\tau = 0$ when choosing the best continuation.

To measure Hits@K Accuracy, we use ChatGPT (gpt-3.5-turbo; OpenAI (2023)) instead of GPT-4. We set $top\_p = 0.5$ and temperature $\tau = 1.0$ when sampling solutions from ChatGPT.

A.3 Ablation Study

5 shows the ablation study results on the number of tags.

	20 Tags	50 Tags	100 Tags
Date	98.0	98.8	99.2
Formal	63.2	93.2	80.8
Geometric	86.4	87.6	86.0
Logical	98.9	99.5	99.1
Movie	93.6	95.2	94.8
ObjCount	99.6	95.2	99.6
Penguins	99.3	99.3	99.3
ReasonColored	100.0	100.0	100.0
RuinNames	90.8	93.6	95.6
TranslationError	72.8	75.2	72.4
Temporal	98.8	100.0	99.2
WordSort	57.6	56.0	61.6
Average	88.3	**91.1**	90.6

Table 5: Ablation Study: Hits@K Accuracy on BBH tasks using varying number of *fLSA* tags.

[7]All datasets used in the work are under MIT license. Our use of the datasets is consistent with their intended use.