Programming Assignment 3: Topic Modelling

Isak Munther, Daniel Jonsson Bergsten, Melker Rååd, Ylva Eriksson
DAT450 – Natural Language Processing

November 19, 2025

1 Introduction

Topic modelling is a family of probabilistic methods used to uncover hidden thematic structure in large text collections. Latent Dirichlet Allocation (LDA) is a widely used model of this type. LDA assumes that documents are mixtures of latent topics, and that each topic defines a probability distribution over words. By estimating these structures from data, the model can reveal coherent semantic patterns that are not directly observable.

LDA generates each document by first sampling a topic mixture for the document, and then repeatedly selecting a topic and a word according to the model parameters. Inference consists of reversing this generative process: given a corpus, the task is to estimate which topics generated which words. Since the exact posterior distribution is intractable, approximate inference is required.

In this work we use collapsed Gibbs sampling, a Markov chain Monte Carlo method that integrates out the latent topic–word and document–topic distributions and samples only the topic assignment for each word token. This is a standard choice for LDA and works well in practice.

To evaluate topic quality we compute the UMass coherence score, which is based on document-level co-occurrence statistics. Coherence captures how semantically related the top words of a topic are and provides a quantitative way to compare different hyperparameter settings.

2 Background

2.1 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) is a generative probabilistic model for discovering latent thematic structure in text corpora. Each document is represented as a mixture of K topics, and each topic corresponds to a distribution over words. For each document d, the document–topic proportions θ_d are drawn from a Dirichlet prior with hyperparameter α. For each topic k, the topic–word distribution ϕ_k is drawn from a Dirichlet prior with hyperparameter β.

The generative process for LDA is:

1. For each topic $k = 1, \ldots, K$, draw a word distribution $\phi_k \sim \text{Dirichlet}(\beta)$.
2. For each document d:
   (a) Draw a topic mixture $\theta_d \sim \text{Dirichlet}(\alpha)$.
   (b) For each word token n in document d:
      i. Sample a topic assignment $z_{dn} \sim \text{Categorical}(\theta_d)$.
      ii. Sample a word $w_{dn} \sim \text{Categorical}(\phi_{z_{dn}})$.

Given a corpus, the aim is to estimate the latent topic assignments and the underlying topic and document distributions. Since exact inference is intractable, approximate methods such as Gibbs sampling or variational inference are used.

2.2 Collapsed Gibbs Sampling for LDA

Collapsed Gibbs sampling integrates out the latent variables θ_d and ϕ_k, leaving only the topic assignments z_{dn} to be sampled. This reduces variance and simplifies the sampling procedure.

For each word token, the sampler updates z_{dn} according to the conditional posterior

$$p(z_{dn} = k \mid z_{\neg dn}, w) \propto \frac{n_{wk}^{\neg dn} + \beta}{n_k^{\neg dn} + V\beta} \cdot \frac{n_{dk}^{\neg dn} + \alpha}{n_d^{\neg dn} + K\alpha}, \tag{1}$$

where

- $n_{wk}^{\neg dn}$ is the number of times word w is assigned to topic k, excluding the current token,
- $n_k^{\neg dn}$ is the total number of word tokens assigned to topic k,
- $n_{dk}^{\neg dn}$ is the number of tokens in document d assigned to topic k,
- $n_d^{\neg dn}$ is the total number of tokens in document d,
- V is the vocabulary size.

After sampling, the topic–word and document–topic distributions are reconstructed from the final counts.

2.3 UMass Coherence

Topic coherence measures how semantically meaningful the top words of a topic are. The UMass coherence score is based on document co-occurrence counts in the corpus. For a topic with top words $(w_1, \ldots, w_M)$, the score is

$$C_{\text{UMass}} = \sum_{m=2}^{M} \sum_{l=1}^{m-1} \log \frac{D(w_m, w_l) + 1}{D(w_l)},$$

where $D(w)$ is the number of documents containing word w, and $D(w_m, w_l)$ is the number of documents in which both words appear. Higher scores indicate more coherent topics.

3 Method

3.1 Implementation Details

LDA is implemented using collapsed Gibbs sampling. Each document is represented as a sequence of integer word IDs. The sampler maintains three NumPy-based count structures:

- a word–topic count matrix n_{wk},
- a document–topic count matrix n_{dk},
- a topic count vector n_k.

Each word token is initially assigned a random topic, and all count matrices are populated accordingly. During each iteration, every word token is resampled by temporarily removing its contribution from the counts, computing the collapsed Gibbs conditional, and drawing a new topic. After the final iteration, the topic–word distributions ϕ_k are obtained by normalising the rows of n_{wk} with the prior β.

3.2 Hyperparameters and Settings

Four configurations of the LDA model are evaluated:

- $K \in \{10, 50\}$,
- $(\alpha, \beta) \in \{(0.1, 0.1), (0.01, 0.01)\}$,
- 100 Gibbs sampling iterations for each configuration.

Each run starts from a random initialisation, and the final sample after 100 iterations is used for evaluation.

3.3 Evaluation Protocol

Topics are evaluated both qualitatively and quantitatively:

- Top words: the 20 most probable words in each topic are inspected.
- UMass coherence: computed using the top $M = 20$ words for each topic.

In addition, summary plots show the average coherence and the distribution of topic coherence scores.

4 Experiments

4.1 Experimental Design

The collapsed Gibbs sampler is run for all four combinations of K and (α, β) using the same preprocessed corpus. Each run produces:

- topic–word distributions,
- top-word lists,
- UMass coherence scores for all topics.

This setup makes it possible to compare how the number of topics and prior strength influence topic quality and coherence.

4.2 Runtime and Mixing

Each run completes in a few minutes on a standard CPU. The per-token sampling loop dominates the runtime. The topic–word distributions and coherence values stabilise well within the 100 iterations used here, and the final iteration is taken as the representative sample for analysis.

5 Results

5.1 Quantitative Results: Coherence Scores

Table 1 reports the average UMass coherence across topics for each hyperparameter setting. Since higher coherence (less negative values) is better, the best overall configuration is $K = 10$, $\alpha = \beta = 0.1$. Increasing the number of topics to $K = 50$ lowers the average coherence and increases the spread of topic quality.

Setting	K	α	β	UMass coherence (mean $\pm$ std)
Run 1	10	0.1	0.1	-350.9 ± 40.5
Run 2	50	0.1	0.1	-388.2 ± 68.0
Run 3	10	0.01	0.01	-366.9 ± 66.7
Run 4	50	0.01	0.01	-386.3 ± 66.5

Table 1: UMass coherence for different hyperparameter settings. Higher values (less negative) indicate more coherent topics.

Overall, fewer topics ($K = 10$) give higher average coherence and slightly more stable topics, while the sparser prior $\alpha = \beta = 0.01$ tends to produce a wider spread of topic qualities compared to $\alpha = \beta = 0.1$.

5.2 Qualitative Results: Top Words per Topic

Qualitative inspection of the top words supports the coherence results. For $K = 10$ and $\alpha = \beta = 0.1$, several topics are clearly interpretable, such as a Middle East / politics topic and a religion topic:

Topic (for $K = 10$, $\alpha = \beta = 0.1$)	Top 10 words
"Middle East / politics"	government, state, rights, israel, group, jewish, israeli, jews, believe, years
"Religion"	god, jesus, believe, christian, christians, church, law, things, true, well

Table 2: Example topics and their top words for $K = 10$, $\alpha = \beta = 0.1$.

For $K = 50$, many topics still look meaningful, but there are more narrow or noisy topics with mixed vocabularies, which is consistent with the lower average coherence and larger variance. The full top-word tables for all configurations are generated by the code.

5.3 Visualisations

Figure 1 shows the average coherence for each configuration, and Figure 2 compares the distribution of topic-wise coherence scores for $K = 10$ and $K = 50$. The boxplots highlight that $K = 50$ yields a wider range of topic qualities, with a few highly coherent topics but many low-coherence ones.

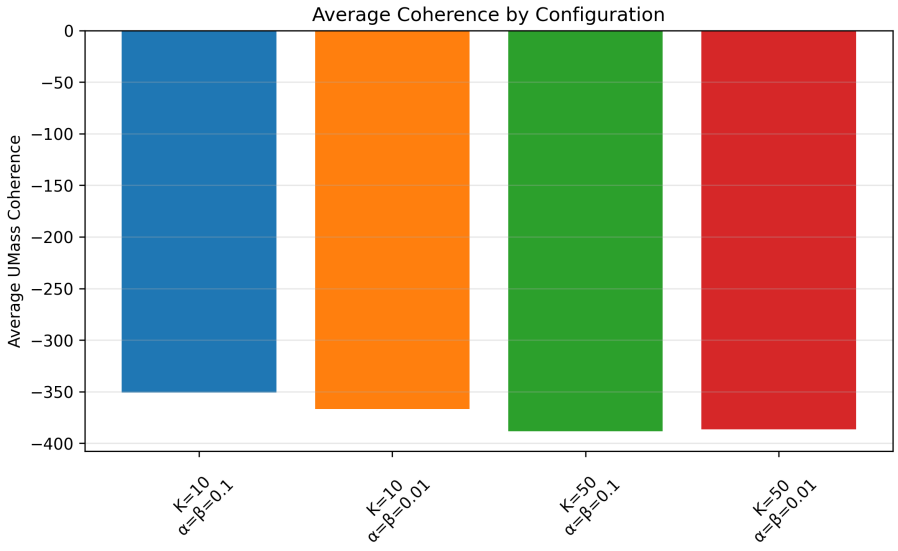

Figure 1: Average UMass coherence by configuration.

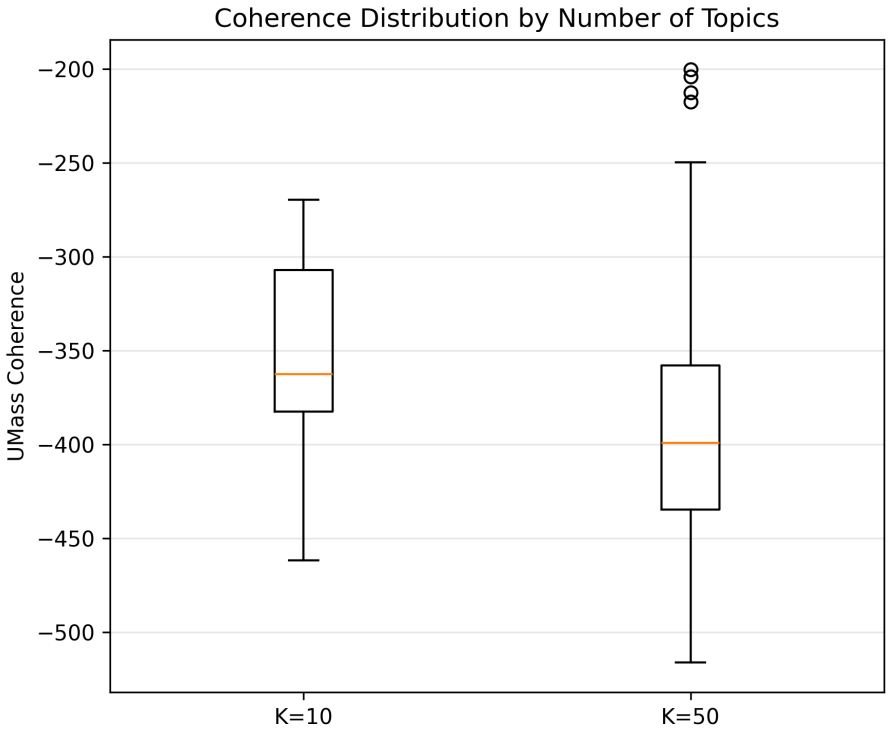

Figure 2: Distribution of UMass coherence scores for $K = 10$ and $K = 50$.

6 Discussion

The results show clear patterns in how the LDA model behaves under different hyperparameter settings. The coherence scores indicate that models with fewer topics generally perform better: both configurations with $K = 10$ achieve higher average coherence than their $K = 50$ counterparts. Increasing the number of topics leads to more variability, with a mix of highly coherent topics and many low-quality ones, as seen in the wider spread in the coherence distributions. The effect of the Dirichlet prior is smaller but still noticeable: the sparser setting ($\alpha = \beta = 0.01$) tends to produce slightly sharper yet less stable topics than $(0.1, 0.1)$.

Topics with high coherence usually correspond to semantically meaningful themes. For example, the best topics for $K = 10$ include well-defined clusters such as religion or Middle East politics. In contrast, the lower-coherence topics, especially for $K = 50$, often contain mixed vocabularies or overly narrow themes that do not form a clear topic. This matches the tendency of LDA to generate fragmented or redundant topics when K is set too high.

Some common failure modes appear across settings. Several topics are close duplicates of others with slightly shifted word distributions, while others are too broad or collect frequent but uninformative words. These issues reflect the model's sensitivity to corpus size, preprocessing choices, and hyperparameters.

With more time, several improvements could be explored. Running the Gibbs sampler for more than 100 iterations might help the model converge more fully. Better preprocessing, such as lemmatisation, phrase detection, or more refined stop-word filtering, could improve topic quality. Finally, experimenting with asymmetric or non-symmetric priors could encourage more realistic document–topic distributions and reduce redundant topics.

7 Conclusion

This project implements Latent Dirichlet Allocation using collapsed Gibbs sampling and applies it to a preprocessed subset of the 20 Newsgroups corpus. The implementation integrates out the document–topic and topic–word distributions and updates only the latent topic assignments. Four hyperparameter configurations are evaluated by varying the number of topics and the Dirichlet priors.

The experiments show clear differences in model behaviour across settings. Models with fewer topics ($K = 10$) consistently produce higher UMass coherence and more interpretable topics, whereas increasing the number of topics to $K = 50$ leads to lower average coherence and greater variability, with a mixture of coherent and weak topics. The choice of prior also influences sparsity: the smaller prior ($\alpha = \beta = 0.01$) yields sharper but less stable topics than $(0.1, 0.1)$.

Overall, the results highlight the sensitivity of LDA to both the number of topics and the Dirichlet hyperparameters. Coherence scores line up well with qualitative inspection, which suggests that coherence is a useful indicator of topic quality. Possible extensions include more extensive hyperparameter tuning, longer sampling runs, or alternative priors to encourage clearer topic separation.

Source Code

The implementation is submitted as two Python files:

- `download_20newsgroups.py`: downloads the 20 Newsgroups corpus using `scikit-learn` and saves it as `data/20newsgroups_raw.json`.
- `main.py`: loads and preprocesses the corpus, builds the vocabulary, runs collapsed Gibbs sampling for all hyperparameter settings, computes UMass coherence, and generates the LaTeX tables and result plots.

To reproduce the experiments, first run `download_20newsgroups.py` to create the JSON file in the `data/` directory, and then run `main.py` to train the models and produce the results.