Algorithm Implementation Details

Overview and Rationale

We used two modeling techniques- Latent Dirichlet Allocation(LDA) and BERTopic, to extract themes from the INK posts. While LDA is the classical approach, BERTopic leverages modern deep learning(via BERT embeddings) to capture semantic meaning beyond word occurrence. This allowed us to evaluate both interpretability and performance.

After preprocessing our data, the first algorithm we worked on is LDA.

Latent Dirichlet Allocation

Theoretical Background

LDA is a generative probabilistic model designed to uncover latent themes or topics in large collections of text documents(GeeksforGeeks, 2024).It functions as a reverse-engineering algorithm: much like identifying ingredients in a dish, LDA attempts to infer which topics and word groupings were likely used to generate each document.

It operates on two core assumptions:

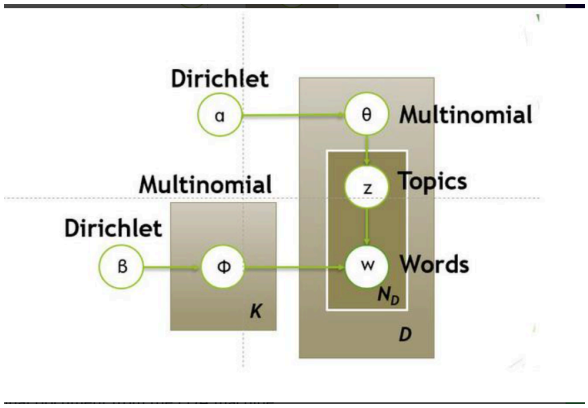
- Every document is a mixture of topics.
- Every topic is a mixture of words

The model uses a generative process, where it assumes that for each document, a distribution over topics is selected, and for each word in the document, a topic is sampled from this distribution. Then a word is chosen from the corresponding topic's word distribution.

In practice, LDA treats documents as a "bag of words" meaning it disregards word order and focuses purely on word frequency and co-occurrence. These frequencies are captured in a document term matrix, where rows represent documents and columns represent words. Each cell shows how often a word appears in a given document.

To infer the hidden topic structure from the observed data, LDA uses Collapsed Gibbs sampling, a method that iteratively estimates distributions of topics and words by repeatedly sampling topic assignments for each word in the corpus, based on current estimates of surrounding topic-word and document-topic distributions. Over many iterations, Gibbs sampling converges towards stable topic assignments.

Mathematically, this is represented by the equation:



$$P(W, Z, \theta, \varphi, \alpha, \beta) = \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{i=1}^{K} P(\varphi_i; \beta) \prod_{t=1}^{N} P(Z_{j,t}|\theta_j) P(W_{j,t}|\varphi_{Z_{j,t}})$$

$D: Number\ of\ Documents$
$N_d: Number\ of\ words\ in\ a\ given\ document$
$\beta: dirichlet\ prior\ on\ the\ per-document\ topic\ distribution$
$\alpha: dirichlet\ prior\ on\ the\ per-topic\ word\ distribution$
$\theta_i: topic\ distribution\ for\ document\ i$
$\varphi_k: word\ distribution\ for\ topic\ k$
$z_{ij}: topic\ for\ the\ j-th\ word\ in\ document\ i$
$w_{ij}: specific\ word.$

Our Implementation

After our initial data preprocessing, we went on to split our 78 weeks of INK data into a train set, validation test and test set using the 80/20 - 80/20 rule. However, on further research, we found that LDA does not require a traditional validation- test split because it is an unsupervised learning algorithm. Unlike supervised models that need labeled data to learn from and validate against, LDA works directly on the full corpus to uncover latent topic structures without relying on predefined labels or topics.

Upon research we found that the **gensim** library is the most widely used and preferred library for LDA due to its scalability and flexibility(morrisb, 2025). However, all attempts to download it on collab and even on our local computers failed. The error mentioned a dependency problem as the gensim versions available, if installed, would be incompatible with the **numpy, sympy and thinc versions** - other libraries used for our data exploration and processing. After several attempts, we pivoted to using the Sklearn library which also works for LDA.

LDA using the Sklearn Library

To begin, we used Scikit-learn's CountVectorizer to convert our pre-processed INK reflections into a document-term matrix. This matrix forms the foundation on which LDA operates. We applied stricter vectorization parameters(such as max_df = 0.4, min_df = 7,token_pattern = r'\b[^d\W]{3,}\b',Ngram_range = (1,5)) to filter out uninformative words and improve topic coherence:

After vectorization we trained the LDA model using the LatentDirichletAllocation class from Scikit-learn. Key parameters included:

- N_components = 10 which specifies the number of topics to extract.
- Max_iter = 200: allows the algorithm enough iterations to converge to a meaningful topic distribution.
- Batch_size = 128: controls how many documents are processed per interaction in online learning in chunks to make it more memory efficient.
- Doc_topic_prior = 0.1(alpha): A lower alpha value encourages topic sparsity in documents, meaning each document tends to focus on fewer topics.
- Topic_word_prior = 0.01(beta): A low beta value promotes sparsity in topic-word distributions, so each topic is represented by a more distinct set of keywords.

After training, we examined the learned topics by printing out the top 7 keywords for each one and also did the document-topic correlation using the `doc_topic_dist = lda.transform(X)` which showed the distribution of topics in each document in terms of probabilities, and matched each of the 78 weeks(documents) to a main topic. Such as:

```
Document 0:
  Main topic: 1 (Confidence: 0.98)
  Full topic distribution: [0.00208381 0.981245   0.00208381 0.00208441 0.00208402 0.00208391
 0.00208344 0.00208338 0.00208398 0.00208425]
Document 1:
  Main topic: 1 (Confidence: 0.80)
  Full topic distribution: [7.04407893e-04 7.99645546e-01 7.04438662e-04 7.04389645e-04
 1.94719459e-01 7.04417112e-04 7.04322843e-04 7.04270557e-04
 7.04334464e-04 7.04413018e-04]
```

Model Evaluation and Hyper Parameter Tuning: A peek into our challenges and lessons

Our model evaluation was in 3 forms:

- Coherence Score: measures the semantic similarity between high-scoring words in each topic and helps ensure that the words within each topic are meaningful and related.(Tiya Vaj, 2023). We used the cv coherence which measures coherence in a range of 0.0 to 1.0. Gensim library, again, had a simple and implicit way of calculating coherence, but due to the dependency errors, we had to find manual ways of calculating it from scratch. Tools like chatgpt and deepseek helped.
- Perplexity score: assess how well our model generalizes to new unseen data. Lower perplexity scores usually in the range (300 - 800) indicate better generalization
- Human Evaluation.

The key parameters we used were settled on after numerous hyper parameter tuning. We wrote this method, to test all parameters:

```
for n in [3,5,7, 8, 10, 15, 20]:  # Test different topic counts
    lda = LatentDirichletAllocation(
                                n_components =n,# more topics increase coherence
                                random_state=42,
                                learning_method='online',
                                max_iter=300,
                                batch_size=128,
                                doc_topic_prior=0.05,  # Alpha (lower = sparse topics) the alpha and beta distributions in LDA
                                topic_word_prior=0.005  # Beta (lower = sparse words)
                                )

    lda.fit(X)
    coherence_score = calculate_cv_coherence(lda, vectorizer, processed_text)
    perplexity = lda.perplexity(X)
    print(f"n_components={n}, Coherence={coherence_score:.2f}, Perplexity = {perplexity:.2f}")
```

From these parameters set, we got this:

```
n_components=3, Coherence=0.18, Perplexity = 772.75
n_components=5, Coherence=0.17, Perplexity = 777.79
n_components=7, Coherence=0.21, Perplexity = 879.40
n_components=8, Coherence=0.21, Perplexity = 964.54
n_components=10, Coherence=0.23, Perplexity = 829.21
n_components=15, Coherence=0.22, Perplexity = 1061.97
n_components=20, Coherence=0.22, Perplexity = 1226.70
```

And after changing each parameter and keeping the rest constant at each turn, we realized that n_components = 10(10 topics) always gave a higher coherence and an okay perplexity.

However a coherence of 0.23 is low by most standards, and we sought many ways to optimize it, such as using another library for LDA - the TF-IDF library, which we learned may improve coherence depending on the type of data. Unfortunately it gave even lower scores - 0.17.

We also noticed from the results of the top n words per topic that words like ‘like’, ‘want’ etc, sometimes appeared all too often in the results and they did not contribute much to the semantic meaning of the topics. We decided to write additional stopwords to add to the english stopwords.

```
additional_stopwords = [
    'chale', 'dey', 'go', 'come', 'make', 'wey', 'saa', 'sef',
    'would', 'could', 'should', 'also', 'get', 'many', 'much',
    'etc', 'ok', 'okay', 'yes', 'no', 'yeah', 'nah','hmm','talk'
    'today', 'tomorrow', 'yesterday', 'day', 'week', 'month','like',
    'big','come','need','want','wasnt'
]
en_stop = set(stopwords.words('english')).union(additional_stopwords)
```

However, the coherence scores also decreased, so we took them out and reduced the number of top n words to reduce the possibility of these words appearing much while trying to maintain coherence.

Human Evaluation

Since topic modelling does not actually label the topics, we sent out a google form to a few friends to see what sense they made out of the topic, whether they think they correlate and to give their own labels to it. (links to both the google form and the responses:
https://forms.gle/BXoc4TLtikSWFV6i6
https://forms.gle/xj2Tw4m3xNkGE6gt8)

From their responses and our own judgement, we labeled the ten topics as () and looked at about 10 documents and checked if the topic attached to them was actually representative of much of the document.

Our conclusion: The model did fairly well.

The BERTopic

In addition to LDA, we implemented BERTopic, a modern topic modeling technique that leverages language models and clustering algorithms to uncover meaningful topics. Unlike LDA, which relies on word frequency and bag-of-words assumptions, BERTopic incorporates semantic context using transformer-based embeddings, making it well-suited for short and informal texts like our INK dataset. (Dhanushkumar, 2024)

Our Implementation

After the loading and cleaning of the 78 weeks of data, documents shorter than 10 characters were excluded to ensure quality output.

Key Components of the BERTopic Setup

- Embedding Model**:** we used ‘all-MiniLM-L6-v2’, a fast and lightweight sentence BERT model that generates dense vector representations capturing semantic relationships between reflections.
- Using UMAP(n_neigbors = 5, n_components = 5),we reduced the high-dimensional embeddings to make clustering more effective while preserving local structure in the data.
- We used the HDBSCAN(min_cluster_size=3) to group semantically similar documents. HDBSCAN automatically determines the number of clusters and can identify outliers(documents not strongly related with any topic).
- We customized a countvectorizer with ngram_range =(1,3), min_df = 5, and max_df = 0.75 to filter uninformative terms and phrases in the corpus.
- We used Maximal Marginal Relevance(MMR) with diversity = 0.5 to ensure that each topic’s keywords were both relevant and diverse to reduce redundancy among terms.

After fitting the model, we extracted the list of topics and their associated keywords using the get_topic_info() and get_topic(). For qualitative insights, we also used get_representative_docs() to retrieve examples from the dataset that best represented each topic to make the interpretation more intuitive. We ignored Topic -1, which BERTopic reserves for outlier documents.

A powerful feature of BERTopic is its flexibility in how topics are represented and interpreted. BERTopic allows one to customize the logic used to select representative keywords for each topic and this can improve both the clarity and coherence of topic descriptions. These representations all give different and interesting perspectives on the topics. **Thus we did two representations and that is why you will find two BERTopic code blocks in our code**. The second codeblock used **the keyBERTInspired Representation. ()**

This representation uses techniques from KeyBERT, a keyword extraction method based on BERT embeddings. Instead of picking the most common words in a topic, it:

- Ranks words based on how semantically similar they are to the topic’s overall meaning.
- Helps generate more interpretable and meaningful keywords for each topic.

We found that using keyBERTInspired() gave cleaner and more insightful topic summaries, especially when compared to the default method which sometimes includes generic or repetitive words.
We also considered using OpenAi based representations which generate summaries using GPT models but it required that we pay for it.

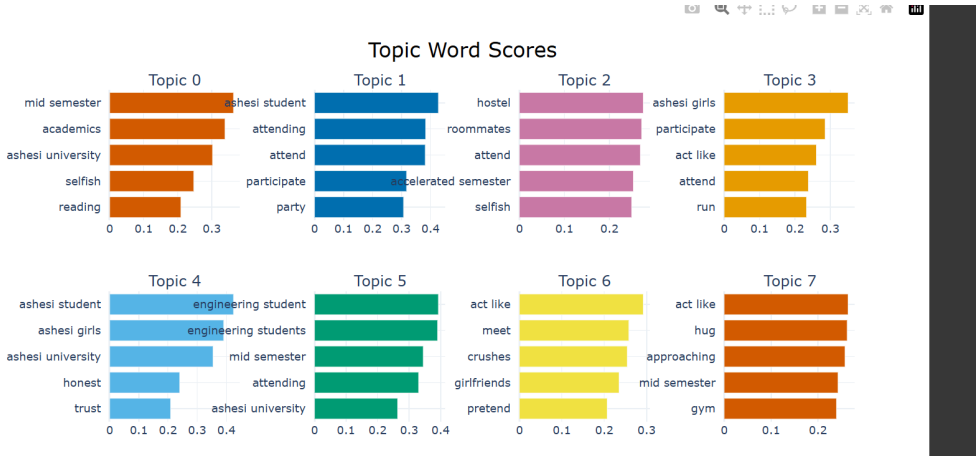
We used word clouds to display the top words of each topic the BERT generated.

Hyper parameter Tuning in BERT

Interestingly, the BERTopic generates slightly different topics even for the same dataset. This variation is due to the HDBSCAN, the clustering algorithm which does not always return the same clusters even with the same data as well as the Embedding Model and the UMAP which randomizes by default.
Thus we couldn't directly fine tune the BERT except by changing these parameters: max_df, min_df and ngram_range.

Evaluating the BERT
We evaluated our BERTopic results using a mix of qualitative inspection and inbuilt representation metrics. A key metric we focused on was the diversity score, which measures how varied the keywords are within each topic. **Our model achieved a 72% diversity score**, indicating that the top keywords for each topic were both semantically relevant and non-redundant. This helped ensure that the topics were not dominated by similar or repetitive words.
We used visualizations which further highlight diversity such as

- Topic word Scores:



This visualization shows the top keywords for each topic, along with their respective importance or score within that topic. This helped us inspect and interpret what each topic is about. This helped us evaluate how interpretable a topic is.

- Similarity Matrix: It is a heatmap which shows the cosine similarity between topics based on their keyword embeddings. It evaluates how similar the topics are to each other and is closely related to diversity.
  Interpretation:
  - Diagonal = 1.0(a topic is 100% similar to itself)
  - Off diagonal values close to 1 suggest overlapping or redundant topics
  - Lower values mean more distinct topics.

Our diversity score of 72% meant that our topics are unique which is a fairly good result. To further evaluate the quality of our topic modeling using BERTopic, we computed the **Document Alignment Score**, which measures how well each document aligns with its assigned topic.(Sajid, 2024).
Our model achieved a score of 0.721, indicating a moderately strong alignment between the documents and their corresponding topics, suggesting that the topics generated by BERTopic meaningfully represent the underlying themes in the data to a good extent.
We wanted to calculate the coherence score as well for BERT but gensim library was needed as well, and an alternative from the octis library was subject to a license.

Human Evaluation
To aid our human evaluation of the BERT, we wrote an HTML code that displayed the top words of each topic and also printed out the documents from the original data that are mapped to that topic.
Since BERT changes the output each time, we did a sample evaluation for one of the BERTopic instances we ran.We could not do another google form in time so we labeled the topics ourselves and checked for coherence with the example documents that were listed. The result is added to the bottom of our code book

Conclusion:
Through this project, our team gained hands-on experience with both traditional and modern topic modeling techniques. While LDA provided a solid baseline with interpretable topics, its limitations with informal text led us to explore BERTopic, which better captured semantic relationships through transformer embeddings. We navigated technical challenges like dependency conflicts and non-deterministic outputs, learning to balance computational trade-offs and evaluation metrics. The iterative process of testing different preprocessing approaches and parameter configurations underscored how data quality and model selection directly impact results.

Beyond the technical work, this project strengthened our collaborative skills as we implemented ethical AI practices for handling student feedback. By combining quantitative metrics with human evaluation, we saw firsthand how topic modeling requires both statistical rigor and thoughtful interpretation. These experiences have given us a practical framework for selecting and refining NLP solutions based on specific use cases and constraints.

Works Cited

DhanushKumar. “Topic Modelling with BERTopic - DhanushKumar - Medium.” *Medium*, Medium, July 2024, medium.com/@danushidk507/topic-modelling-with-bertopic-249095144555.

geeksforgeeks. “Topic Modeling Using Latent Dirichlet Allocation (LDA).” *GeeksforGeeks*, 11 June 2024, www.geeksforgeeks.org/topic-modeling-using-latent-dirichlet-allocation-lda/.

MaartenGr. “BERTopic/Docs/Getting_started/Representation/Representation.md at Master · MaartenGr/BERTopic.” *GitHub*, 2020, github.com/MaartenGr/BERTopic/blob/master/docs/getting_started/representation/representation.md

morrisb. “Compare LDA (Topic Modeling) in Sklearn and Gensim.” *Kaggle.com*, Kaggle, 24 July 2018, www.kaggle.com/code/morrisb/compare-lda-topic-modeling-in-sklearn-and-gensim.

Sajid, Haziqa. “Exploring BERTopic: A New Era of Neural Topic Modeling - Zilliz Blog.” *Zilliz.com*, 28 Mar. 2024, zilliz.com/learn/explore-bertopic-novel-neural-topic-modeling-technique.

Tiya Vaj. “How to Evaluate a Novel Topic Modeling Method. - Tiya Vaj - Medium.” *Medium*, 31 July 2023, vtiya.medium.com/how-to-evaluate-novel-topic-modeling-method-104ad9684428

Other links to documents and videos we used
https://www.analyticsvidhya.com/blog/2023/02/topic-modeling-using-latent-dirichlet-allocation-lda/

GeeksforGeeks. (2024, June 11). *Topic modeling using Latent Dirichlet Allocation (LDA)*.

GeeksforGeeks.

https://www.geeksforgeeks.org/topic-modeling-using-latent-dirichlet-allocation-lda/

Harshit. (n.d.). *Topic Modelling using LDA and LSA with Python Implementation*.

https://www.enjoyalgorithms.com/blog/topic-modelling-using-lda-lsa

Future Mojo. (2022, May 18). *NLP Demystified 9: Automatically Finding Topics in Documents with Latent Dirichlet Allocation* [Video]. YouTube.

https://www.youtube.com/watch?v=9mNV4AwA9QI

Google Colab. (n.d.).

https://colab.research.google.com/github/futuremojo/nlp-demystified/blob/main/notebooks/nlpdemystified_topic_modelling_lda.ipynb