Sentiment Analysis

Mohammadkazem Rajabi
Mohammadkazem.rajabi@unipd.studenti.it

Baharehsadat Khatami
Baharehsadat.khatami@unipd.studenti.it

1. Introduction

Understanding customer sentiment in product reviews is a critical task for companies aiming to assess brand perception and improve user experience. In this project, we explore sentiment classification on user reviews of popular consumer technology products—primarily from Apple and Samsung—using a range of traditional and state-of-the-art NLP methods. The goal is to classify each review as expressing a positive, neutral, or negative sentiment.

While Transformer-based models have become the standard for many NLP tasks, we aim to systematically evaluate how much performance gain they provide compared to simpler and more efficient models such as FastText and recurrent neural networks. We also investigate whether traditional models, when enhanced with attention mechanisms and advanced loss functions like focal loss, can remain competitive.

Our experimental setup includes FastText[10] as a baseline, multiple deep learning models including LSTM[9], GRU[4], and BiLSTM[6] with attention mechanisms, and Transformer-based architectures such as RoBERTa[12], XLNet [13], and DeBERTa-v3[8]. In addition, we apply stratified K-fold cross-validation for robust evaluation, and focal loss to address class imbalance during training.

Beyond supervised classification, we incorporate topic modeling to explore latent semantic themes within the review corpus. We apply four complementary techniques: Latent Dirichlet Allocation (LDA)[3], Latent Semantic Analysis (LSA)[5], Non-negative Matrix Factorization (NMF)[11], and BERTopic[7]. These models help uncover brand-specific thematic structures and offer interpretability that complements the results of our sentiment classification pipeline.

The results show that DeBERTa-v3 consistently outperforms other models in terms of both accuracy and F1-score, while attention-based[1] BiLSTM models demonstrate competitive performance with significantly lower computational cost. Topic modeling further enhances our understanding of the underlying semantic space, providing a holistic view of sentiment and themes in real-world, domain-specific datasets.

2. Dataset

The dataset used in this project is derived from the publicly available Amazon Electronics 5-core dataset provided by Stanford SNAP. It contains millions of user reviews across a wide range of electronic products. We programmatically filtered this dataset to extract reviews specifically related to Apple, Samsung, and Lenovo products.

Brand identification was performed using a set of manually curated keyword lists. Reviews were matched to a brand (Apple, Samsung, or Lenovo) if they contained relevant product keywords such as “iPhone,” “Galaxy,” or “ThinkPad,” while reviews containing irrelevant topics (e.g., “pizza,” “restaurant”) were excluded through a blacklist filtering process.

Each review includes a star rating from 1 to 5. We converted these ratings into sentiment labels using the following mapping: 1–2 stars as **negative** sentiment (-1), 3 stars as **neutral** sentiment (0), and 4–5 stars as **positive** sentiment (1). This produced a labeled sentiment dataset grounded in real user ratings.

To improve neutral class representation—often underrepresented in natural distributions—we supplemented the data with a set of synthetic neutral reviews generated using large language models (LLMs). These synthetic samples were created through prompt engineering to elicit objective, descriptive statements that reflect neutral sentiment. The final dataset contains approximately 17,000 labeled reviews, covering a balanced mix of brands and sentiment classes.

3. Methodology

3.1. Preprocessing

To ensure consistency and reduce noise, we applied a sequence of standard NLP preprocessing techniques to all input texts. These included lowercasing, punctuation removal, and tokenization. Stopwords were removed using the NLTK corpus to reduce sparsity and emphasize semantically meaningful tokens. We deliberately avoided stemming and lemmatization to retain lexical nuance, which is beneficial for pre-trained Transformer-based models.

3.2. Model Architectures

3.2.1 FastText: Lightweight and Efficient Text Classification

FastText, developed by Facebook AI Research, is a shallow neural network model designed for fast and scalable text classification. It enhances traditional word embedding approaches by incorporating **subword-level information**, allowing it to better handle rare, misspelled, or morphologically rich words.

The FastText model consists of the following components:

- **Input Representation:** Each word is decomposed into a set of character n-grams. For instance, the word *where* with $n = 3$ is represented as `<wh, whe, her, ere, re>`, including boundary symbols.

- **Embedding Layer:** Each n-gram is associated with a vector representation. The word vector is obtained by summing its n-gram vectors.

- **Sentence Representation:** A sentence is represented by averaging the vectors of the words it contains.

- **Classifier:** The sentence vector is fed into a linear classifier, typically a softmax layer, to predict the probability distribution over predefined classes.

- **Training Objective:** The model is trained using a hierarchical softmax loss function, optimizing the parameters to predict the correct class labels efficiently.

Although FastText lacks the contextual depth of Transformer-based models, it delivers strong baseline performance and serves as an effective model for low-resource or large-scale sentiment classification applications.

3.2.2 Deep Neural Models

We implemented several deep learning architectures based on Recurrent Neural Networks (RNNs), focusing on LSTM and GRU variants for sentiment classification. All models processed padded token sequences (vocab size: 30,000; max length: 200) to predict one of three sentiment classes.

BiLSTM + Attention. Our baseline model employed a Bidirectional LSTM (128 units), followed by a self-attention mechanism and a second BiLSTM layer (64 units). It used randomly initialized embeddings and was trained with categorical cross-entropy, class weighting, and early stopping.

BiLSTM + GloVe + Focal Loss. We extended the baseline with pre-trained 300D GloVe embeddings (non-trainable) and focal loss to better handle class imbalance. This model achieved strong generalization and was used for final test evaluation.

Other Variants. We also experimented with GRU/BiGRU models (faster but less accurate) and a Conv1D + BiLSTM hybrid to capture local features before temporal modeling.

3.2.3 Transformer-based Models

We employed three Transformer-based language models—RoBERTa, XLNet, and DeBERTa-v3—for fine-tuning using the HuggingFace Transformers library. These models, originally trained on extensive corpora, leverage either masked or permutation-based objectives and have consistently delivered top-tier performance on a range of NLP benchmarks.

RoBERTa refines BERT by discarding the Next Sentence Prediction task and enhancing training with dynamic masking and extended input lengths. XLNet adopts a permutation-based language modeling approach, enabling it to model bidirectional dependencies while preserving the benefits of autoregressive training. DeBERTa-v3 enhances representation learning through disentangled attention and relative position embeddings, making it particularly effective in scenarios with limited data.

In all cases, a linear classification layer was added atop the final encoder output, and the full model was fine-tuned end-to-end on our sentiment dataset. Early stopping was applied to monitor validation loss and reduce overfitting risks.

3.3. Evaluation Results

3.3.1 Transformer Model Performance

To evaluate the effectiveness of pre-trained Transformer architectures in sentiment classification, we fine-tuned five language models: DeBERTa-v3-base, DeBERTa-v3-large, RoBERTa, XLNet, and CardiffNLP[2]. These models have shown strong performance across various NLP tasks, and we assess their suitability for multi-class sentiment classification on our dataset of product reviews.

All models were fine-tuned using a linear classification head on top of the final hidden state representation, with training carried out using early stopping and the AdamW optimizer. We evaluated the models using macro-averaged metrics on a stratified held-out test set. The results are reported in Table 1.

The results show that CardiffNLP achieves the highest overall performance across all metrics, closely followed by DeBERTa-v3-large. Despite being smaller in size, DeBERTa-v3-base exhibits strong precision, suggesting it

Table 1. Test set performance of Transformer-based models.

Model	Accuracy	Precision	Recall	F1-score
CardiffNLP	**0.7598**	0.7524	**0.7598**	**0.7546**
DeBERTa-v3-large	0.7526	**0.7539**	0.7526	0.7532
DeBERTa-v3-base	0.7228	0.7572	0.7228	0.7332
RoBERTa	0.7260	0.7204	0.7260	0.7223
XLNet	0.7088	0.7371	0.7088	0.7180

is a viable option when computational efficiency is a priority. RoBERTa and XLNet deliver reasonable performance but lag behind the DeBERTa family. Interestingly, XLNet achieves high precision but struggles with recall, possibly due to overfitting or instability during fine-tuning. These results reinforce the effectiveness of modern Transformers for sentiment classification, particularly those using disentangled attention mechanisms and large-scale pretraining.

3.4. Deep Learning Model Performance

We implemented and compared a range of deep learning models to perform sentiment classification on our multiclass dataset. These include baseline BiLSTM architectures as well as enhanced models using convolutional layers, attention mechanisms, pretrained word embeddings (GloVe), and focal loss to address class imbalance.

All models were trained on the same dataset using either stratified train-test splits or 5-fold cross-validation (CV), and evaluation was based on test accuracy and macro-averaged F1-score.

Table 2. Test set performance of deep learning models.

Model	Accuracy	F1-score
BiLSTM+Attention+GloVe+ Focal Loss	**0.7019**	0.6500
BiLSTM + Attention + GloVe	**0.7050**	0.6600
GRU + GloVe	**0.7084**	**0.7099**
BiLSTM + Attention	**0.7039**	0.6800
Conv1D + BiLSTM	0.6817	0.6820
Vanilla BiLSTM	0.6674	0.6696
GRU	0.6348	0.6408

Model descriptions: The **Vanilla BiLSTM** model is a baseline architecture consisting of two stacked Bidirectional LSTM layers with dropout and dense output. The **BiLSTM + Attention** model enhances this by adding a self-attention mechanism between the LSTM layers to help the model focus on informative tokens. The **GRU + GloVe** model uses pretrained GloVe word embeddings and replaces LSTM layers with GRU units, improving generalization. The **GRU** model uses GRU layers without GloVe and demonstrates weaker performance, emphasizing the benefit of pretrained embeddings.

The **Conv1D-BiLSTM** model introduces a 1D convolutional layer before the BiLSTM to extract local n-gram patterns, followed by global max pooling and classification layers. Lastly, the **BiLSTM + Attention + GloVe + Focal Loss** model combines the strengths of pretrained embeddings, attention mechanisms, and focal loss to effectively handle class imbalance and delivers the most balanced overall performance across all classes.

3.5. FastText Sentiment Classification

Preprocessing and Format: We used the same dataset of 17,000 reviews labeled across three sentiment classes (negative, neutral, positive). Reviews were converted to FastText format by prefixing each entry with `__label__NEG`, `__label__NEU`, or `__label__POS`.

Training and Evaluation: The dataset was split using a stratified 80/20 train-test split. The model was trained using FastText's `supervised` mode with default parameters.

Table 3. Class-wise performance of FastText model

Class	Precision	Recall	F1-score
Negative (NEG)	0.6750	0.5807	0.6243
Neutral (NEU)	0.4803	0.4699	0.4751
Positive (POS)	0.7444	0.8177	0.7794

Overall Evaluation:

- **Macro-Averaged F1-score:** 0.6263
- **Overall Accuracy:** 66.15%

Discussion: Despite its straightforward architecture and fast training time, FastText performed surprisingly well. It showed the highest confidence when identifying positive reviews, but—like the more complex deep learning and Transformer-based models—it struggled most with the neutral class. Given its minimal resource requirements, FastText remains a compelling choice, especially for scenarios where scalability or computational efficiency is a priority.

3.6. Shallow Neural Networks and Classical ML Baselines

To complement our evaluation of advanced deep learning models, we also examined the performance of shallow neural networks and classical machine learning baselines. These included simplified GRU-based architectures and traditional classifiers using TF-IDF features. While the shallow models demonstrated modest gains from bidirectionality, classical approaches generally underperformed. Overall, this comparison highlights the limitations of shallow and feature-based methods in capturing the nuanced structure of sentiment-laden text.

As shown in Table 4, shallow neural models and classical methods underperform compared to deeper architectures.

Table 4. Performance of shallow neural networks and classical ML baselines.

Model	Accuracy	F1
GRU (1 layer)	0.62	0.58
BiGRU (shallow)	0.66	64
Logistic Regression (TF-IDF)	0.59	0.55
Random Forest and Logistic Regression	0.64	0.58
Decision Tree (TF-IDF)	0.53	0.49

4. Topic Modeling Exploration

To enrich our understanding of the review dataset beyond sentiment polarity, we conducted an unsupervised topic modeling analysis. This allowed us to extract underlying themes, user concerns, and contextual patterns that frequently emerge in tech-related reviews. We employed four widely-used topic modeling approaches: **Latent Dirichlet Allocation (LDA), Latent Semantic Analysis (LSA), Non-negative Matrix Factorization (NMF), and BERTopic**.

4.1. Preprocessing and Input Representation

All reviews were preprocessed by converting to lowercase, removing punctuation, and filtering out English stopwords. Lemmatization was not applied.

Depending on the algorithm, two types of vectorization were used: Text preprocessing involved lowercasing, punctuation removal, and stopword filtering. For **LSA** and **NMF**, we used a TF-IDF matrix to emphasize informative terms. **LDA** relied on a bag-of-words representation suited to its probabilistic nature. **BERTopic** used dense embeddings from the `all-MiniLM-L6-v2` model, followed by HDBSCAN clustering and topic extraction using class-based TF-IDF.

4.2. Latent Dirichlet Allocation (LDA)

LDA [3] is a generative probabilistic model that represents documents as mixtures of topics, where each topic is a distribution over words. In our analysis, we trained the LDA model with 10 topics and visualized the top keywords per topic. The results revealed coarse-grained themes such as: we report the top 10 most probable words per topic in Table 5.

4.3. Latent Semantic Analysis (LSA)

LSA [5] is a dimensionality reduction technique based on Singular Value Decomposition (SVD) of the TF-IDF matrix. LSA captures linear substructures and latent relationships in term usage. Despite its mathematical simplicity, the LSA topics were less coherent compared to LDA, likely due to its lack of probabilistic modeling.

Table 5. Top 8 keywords per topic extracted using LDA.

Topic	Top Keywords
Topic 1	cable, usb, adapter, sound, iphone
Topic 2	laptop, lenovo, windows, mouse, computer
Topic 3	drive, card, usb, gb, hard
Topic 4	support, router, wireless, device, wifi
Topic 5	keyboard, keys, key, typing, bluetooth
Topic 6	tv, samsung, sound, player, quality
Topic 7	one, would, battery, samsung, get
Topic 8	case, ipad, screen, like, cover

Table 6. Top 8 topics extracted using Latent Semantic Analysis (LSA). Each row lists the top keywords per topic.

Topic	Top Keywords
Topic 1	case, ipad, cover, keyboard, stand, protection, fits
Topic 2	drive, keyboard, macbook, pro, mac, windows, laptop
Topic 3	tv, samsung, screen, hdmi, player, quality, remote
Topic 4	keyboard, keys, bluetooth, mouse, wireless, key, logitech
Topic 5	cable, tv, works, apple, ipad, usb, samsung, charge
Topic 6	tablet, galaxy, samsung, screen, tab, protector, note
Topic 7	case, keyboard, cable, galaxy, samsung, tablet, tab, fit
Topic 8	macbook, great, pro, product, apple, price, perfectly

4.4. Non-negative Matrix Factorization (NMF)

NMF [11] factorizes the TF-IDF matrix into two lower-rank non-negative matrices. It results in additive topic representations, making the output interpretable and often more sparse. The topics extracted from NMF included:

Table 7. Top 9 topics extracted using Latent Semantic Analysis (NMF). Each row lists the top keywords per topic.

Topic	Top Keywords
Topic 1	Samsung, amazon, problem, support, screen
Topic 2	Ipad, Cover, Case, protection, Leather, Screen, laptop
Topic 3	Hard, Drive, SSD, external, windows, computer
Topic 4	Laptop, Lenovo, Thinkpad, battray, fits, life
Topic 5	Charger, charge, power, adaptor, iphone, port
Topic 6	tablet, galaxy, samsung, note, greate, samsung tab
Topic 7	keyboar, logitech, mouse, wireless, bluetooth,
Topic 8	Cable, HDMI, Monitor, connect , adaptor
Topic 9	sound, quality, headphones, speaker , music, volume

4.5. BERTopic: Transformer-Based Topic Modeling

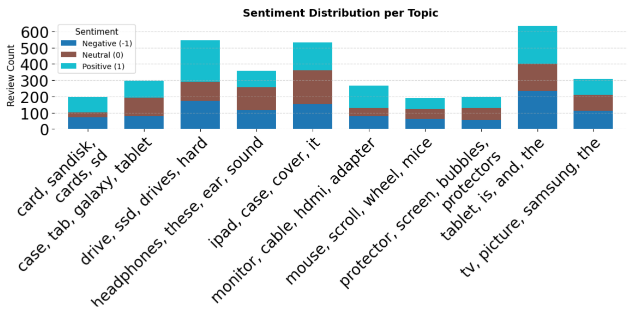
BERTopic [7] is a powerful technique for topic modeling that combines BERT-based sentence embeddings, UMAP for dimensionality reduction, and HDBSCAN for clustering. Unlike traditional bag-of-words approaches, it captures semantic similarity between texts using contextual embeddings.

The model first embeds documents using Sentence-BERT, then reduces dimensionality with UMAP to make clustering more efficient. Clusters are formed using HDBSCAN, and representative words are extracted using a class-based variant of TF-IDF, further improved with KeyBERT for more meaningful keywords. The top keywords from the most prominent topics in our dataset are shown in Table 8.

Table 8. Top 8 topics extracted using BERTopic

Topic	Top Keywords
Topic 1	ipad, case, cover, stand, leather, pen
Topic 2	headphones, ear, these, sound, headset,
Topic 3	card, sandisk, sd, cards, reader, class,
Topic 4	charge, charger, charging, phone, usb, batteray
Topic 5	mouse, scroll, wheel, logitech, mice,
Topic 7	keyboar, logitech, mouse, wireless, bluetooth,
Topic 8	camera, lens, zoom, canon, video, focus, picture

To better understand the relationship between discovered topics and user sentiment, we analyzed how sentiment labels are distributed across the top BERTopic clusters. This gives insight into which topics are associated with more positive, neutral, or negative opinions.



5. Conclusion

In this study, we conducted a comprehensive sentiment analysis on technology product reviews, focusing primarily on brands such as Apple and Samsung. We experimented with a diverse set of modeling techniques, ranging from classical machine learning baselines to deep learning architectures and modern transformer-based language models. Our pipeline also included an extensive topic modeling analysis to uncover the latent thematic structure in user opinions.

Through rigorous experimentation, we observed that traditional approaches such as Logistic Regression or Decision Trees struggled to model sentiment intricacies, particularly in neutral and context-dependent reviews. Shallow recurrent models like single-layer GRUs showed some promise but lacked the expressive power required for nuanced sentiment classification.

Advanced architectures such as BiLSTM with attention mechanisms, models enriched with GloVe embeddings, and focal loss-based training significantly improved performance. Among these, the best-performing deep learning model was the BiLSTM + Attention + GloVe + Focal Loss, achieving an accuracy of 70% and demonstrating robustness in handling imbalanced and noisy labels.

Transformer models such as **DeBERTaV3-large** and **CardiffNLP-TweetEval** achieved the highest accuracy and F1 scores, emphasizing the strength of pre-trained contextual embeddings. Topic modeling techniques including **LDA**, **NMF**, and **BERTopic** revealed key user concerns, with BERTopic and NMF producing more coherent and brand-relevant topics than LDA. These results demonstrate the importance of high-quality data, model depth, and thorough evaluation in building reliable sentiment analysis pipelines for applications like product feedback mining and market analysis.

As an extension of our previous project, we initially applied sentiment classification to a dataset of **8,000** manually labeled Reddit comments on the 2024 U.S. election. While this dataset enabled transformer models to achieve about **96% accuracy**, it was skewed toward strongly polarized sentiments, with few neutral examples—many of which were synthetically generated.

In contrast, our current work focused on a more diverse and realistic dataset of user reviews. This required significant effort to ensure a balanced sentiment distribution, enabling a deeper and more practical evaluation of model performance on naturally ambiguous real-world opinions.

Model	Validation Accuracy	Test Accuracy	F1-Score
DistilBERT	87.1%	91.6%	91.4%
RoBERTa-base	88.7%	90.4%	90.5%
RoBERTa-large	98.2%	97.5%	96.6%
DeBERTa-v3-small	88.8%	91.0%	91.0%
DeBERTa-large	82.0%	89.3%	87.0%

Table 9. Comparison of model performance across different architectures on the U.S. Election 2024 dataset.

In addition to transformer models, we also evaluated custom deep learning architectures on the U.S. Election 2024 dataset. A Bidirectional LSTM model with GloVe embeddings and an attention mechanism achieved a strong test accuracy of 94.3%, outperforming the GRU-based variant, which reached 88.0%. These results demonstrate that, when supported by quality embeddings and attention mechanisms, recurrent models can still offer competitive performance on clean and well-separated sentiment datasets. The results of these models are summarized in Table 10.

Deep Learning Model	Test Accuracy
BiLSTM + GloVe + Attention	94.3%
GRU + GloVe + Attention	88.0%

Table 10. Performance of deep learning models on the U.S. Election 2024 dataset.

Work Contribution and Time Breakdown

Team Members: Mohammad and Bahar
Below is a breakdown of the main activities and the time spent by each team member. Total effort is approximately **34 hours for Mohammad** and **36 hours for Bahar**, in line with course guidelines.

Task	Mohammad (hrs)	Bahar (hrs)
Dataset Collection and Labeling		
Reddit dataset (USA Election) creation + labeling	5	5
Amazon reviews filtering and new dataset prep	2	1
Manual labeling + synthetic neutrals	2	2
Sentiment Classification		
Transformer models (DeBERTa, RoBERTa, XLNet, CardiffNLP)	7	5
Deep learning models (LSTM, GRU, BiLSTM + Attention + GloVe)	8	5
FastText implementation + evaluation	3	2
Classical ML models (LogReg, Decision Tree, RF)	2	4
Topic Modeling		
LDA, LSA, NMF implementations + analysis	5/5	4
BERTopic analysis + figures	2	3
Topic interpretation and LaTeX tables	1	1
Report Writing and Submission		
LaTeX formatting, tables, figures	2	2
Writing (intro, methods, results, conclusion)	3	3
Final edits, polishing, GitHub prep	1	0
Total Hours	**43**	**37**

Infrastructure and Code: For the custom dataset (U.S. Election), we performed all training and fine-tuning on Google Colab using GPUs. For the large-scale Apple-Samsung dataset (approx. 70k reviews), we utilized the university compute cluster. All models, log files, and evaluation outputs are documented and publicly accessible in the ***GitHub repository***.

Note: *Amir Sadeghi* was another teammate who contributed to the labeling of the USA Election dataset. He completed this course in the previous semester and agreed to help with initial annotation efforts. Approximately **9 hours** of the Reddit dataset labeling was done by him.

Appendix: Mathematical Foundations

Recurrent Neural Networks

LSTM: Long Short-Term Memory

LSTMs introduce gating mechanisms to control information flow through time. The main equations governing an LSTM cell are:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad \text{(Forget Gate)} \quad (1)$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad \text{(Input Gate)} \quad (2)$$
$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C) \quad \text{(Candidate Memory)} \quad (3)$$
$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \quad \text{(Cell State Update)} \quad (4)$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad \text{(Output Gate)} \quad (5)$$
$$h_t = o_t \cdot \tanh(C_t) \quad \text{(Hidden State)} \quad (6)$$

GRU: Gated Recurrent Unit

GRUs simplify LSTMs by using only reset and update gates:

$$z_t = \sigma(W_z[h_{t-1}, x_t] + b_z) \quad \text{(Update Gate)} \tag{7}$$
$$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r) \quad \text{(Reset Gate)} \tag{8}$$
$$\tilde{h}_t = \tanh(W_h[r_t \cdot h_{t-1}, x_t] + b_h) \tag{9}$$
$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t \tag{10}$$

Transformer Models

Self-Attention Mechanism

Transformers replace recurrence with self-attention. Given query Q, key K, and value V matrices, attention is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{11}$$

Where:

- Q, K, V are learned linear projections of input tokens.
- d_k is the dimension of the key vectors (used for scaling).

BERT and Variants

BERT is pre-trained using:

- **Masked Language Modeling (MLM)**: Predicts randomly masked tokens.
- **Next Sentence Prediction (NSP)**: Determines sentence continuity.

RoBERTa removes NSP and trains longer with more data. **DistilBERT** is a smaller, faster version of BERT via knowledge distillation.

Topic Modeling Algorithms

Latent Dirichlet Allocation (LDA)

LDA models documents as mixtures of topics, where each topic is a distribution over words.

Generative Process:

1. For each document d:
   (a) Draw topic proportions $\theta_d \sim \text{Dirichlet}(\alpha)$.
   (b) For each word:
      i. Draw a topic $z \sim \text{Multinomial}(\theta_d)$.
      ii. Draw word $w \sim \text{Multinomial}(\phi_z)$.

Joint Probability Objective:

$$P(W, Z, \Theta, \Phi | \alpha, \beta) = \prod_{d=1}^{D} P(\theta_d | \alpha) \prod_{n=1}^{N_d} P(z_{d,n} | \theta_d) P(w_{d,n} | \phi_{z_{d,n}})$$

Posterior inference is approximated using variational inference or Gibbs sampling.

Latent Semantic Analysis (LSA)

LSA uses Singular Value Decomposition (SVD) on the term-document matrix $A \in \mathbb{R}^{m \times n}$:

$$A = U\Sigma V^\top$$

Where:

- U: term-topic matrix
- Σ: singular values
- V: document-topic matrix

To reduce noise, only top k singular values are kept:

$$A_k = U_k \Sigma_k V_k^\top$$

Non-Negative Matrix Factorization (NMF)

NMF factorizes a non-negative matrix A into:

$$A \approx WH$$

Where:

- $W \in \mathbb{R}^{m \times k}$: term-topic matrix
- $H \in \mathbb{R}^{k \times n}$: topic-document matrix

Objective Function:

$$\min_{W,H \geq 0} \|A - WH\|_F^2$$

NMF provides additive and interpretable topic representations.

BERTopic

BERTopic is a modern topic modeling technique that combines transformer-based embeddings with clustering algorithms to discover coherent topics in text. It starts by embedding each document using Sentence-BERT (SBERT), which captures the semantic meaning of text beyond traditional bag-of-words methods.

Since these embeddings are high-dimensional, BERTopic applies Uniform Manifold Approximation and Projection (UMAP) for dimensionality reduction, preserving local and global structure better than PCA or t-SNE. The reduced vectors are then clustered using HDBSCAN, a density-based algorithm that handles clusters of varying size and shape.

Topic representations are extracted from these clusters using a modified TF-IDF that treats each cluster as one aggregated document. The weight of a term t in cluster c is computed as:

$$W_{t,c} = tf_{t,c} \cdot \log\left(1 + \frac{A}{tf_t}\right) \tag{12}$$

To further improve interpretability, BERTopic integrates **KeyBERT**, which uses contextual embeddings from BERT to identify the most representative keywords in each topic, beyond frequency-based selection.

For dynamic corpora, BERTopic also supports temporal analysis. It first builds global topic representations and then computes local topic importance at each timestep i using:

$$W_{t,c,i} = tf_{t,c,i} \cdot \log\left(1 + \frac{A}{tf_t}\right) \tag{13}$$

Overall, BERTopic leverages the strengths of neural embeddings, dimensionality reduction, and flexible clustering to produce interpretable, context-aware topic models.

FastText

The FastText model extends the standard Skip-Gram architecture by representing each word w as a bag of character n-grams. Specifically, a word w is represented by the set $\mathcal{G}_w$ of its character n-grams. The word itself is also included in this set. Each $g \in \mathcal{G}_w$ is associated with a vector $\mathbf{z}_g$.

Word Vector Construction

Each word vector $\mathbf{u}_w$ is constructed by summing the vectors of its n-grams:

$$\mathbf{u}_w = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g$$

This composition allows the model to learn vector representations for rare or even unseen words based on their subword units.

Training Objective

Let $\ell(x) = \log(1 + e^{-x})$ be the logistic loss function. FastText optimizes the following objective over the training corpus:

$$\sum_{t=1}^{T} \left[\sum_{c \in \mathcal{C}_t} \ell\left(s(w_t, w_c)\right) + \sum_{n \in \mathcal{N}_{t,c}} \ell\left(-s(w_t, n)\right) \right]$$

Where:

- T: total number of training words
- w_t: the center (target) word at position t
- $\mathcal{C}_t$: the set of context words for w_t
- $\mathcal{N}_{t,c}$: the set of negative samples for context position c
- $s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^\top \mathbf{v}_c$: the scoring function, with

A.4 Subword Example

For the word `where` with $n = 3$, the n-grams might include:

$$\mathcal{G}_{\texttt{where}} = \{\texttt{<wh}, \texttt{whe}, \texttt{her}, \texttt{ere}, \texttt{re>}, \texttt{<where>}\}$$

The word vector $\mathbf{u}_{\texttt{where}}$ is obtained by summing the embeddings of these 6 elements.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Francesco Barbieri, Luis Espinosa Anke, and Jose Camacho-Collados. Tweeteval: Unified benchmark and comparative evaluation for tweet classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1644–1650, 2020.

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

[4] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

[5] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[6] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[7] Maarten Grootendorst. Bertopic: Neural topic modeling with class-based tf-idf. `https://github.com/MaartenGr/BERTopic`, 2022.

[8] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2021.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 427–431, 2017.

[11] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[12] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[13] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, 2019.