NATIONAL INSTITUTE OF TECHNOLOGY MEGHALAYA

Comparative Analysis of Embedding Techniques

for

Sentiment Analysis on Customer Reviews and Analysis of Factors Impacting Customer Satisfaction

- Samiksha Deb (B22CS029)
- Vanshika Sarraf (B22CS037)

Under the Supervision of

Dr. Thoudam Doren Singh
Assistant Professor, Department of Computer Science and Engineering
National Institute of Technology Meghalaya

1. Introduction

This project extends our previous sentiment analysis work by conducting a systematic comparison of various embedding techniques and transformer-based models for sentiment analysis and topic modeling on customer reviews. We evaluate seven different approaches on a large-scale dataset of 200,000 Amazon Fine Food Reviews, analyzing their performance, scalability, and interpretability.

1.1 Project Evolution

In our minor project, we established a baseline using Bag of Words (BoW) representations combined with Latent Dirichlet Allocation (LDA) for topic modeling and VADER for sentiment analysis on 100,000 reviews. While sentiment analysis reveals how customers feel, topic modeling reveals what they feel about. Combining these two enables us to identify sentiment-bearing themes such as delivery issues, pricing, product quality, etc., which cannot be obtained from sentiment or topics alone.Building upon this foundation, the major project extends the analysis in multiple dimensions:

- Expanded dataset from 100,000 to 200,000 reviews for improved statistical significance
- Implemented traditional embedding approaches (BoW, TF-IDF, Word2Vec, GloVe)
- Incorporated state-of-the-art transformer models (BERT, RoBERTa)
- Evaluated multiple topic modeling techniques (LDA, NMF, K-Means, BERTopic)
- Conducted systematic convergence and coherence analysis for optimal hyperparameter selection
- Performed comprehensive comparative analysis across all approaches

1.2 Research Objectives

The primary objectives of this comparative study are:

- Evaluate performance of different embedding techniques (BoW, TF-IDF, Word2Vec, GloVe) versus transformers (BERT, RoBERTa) for sentiment classification
- Compare topic modeling approaches (LDA, NMF, K-Means, BERTopic) with respect to coherence scores and interpretability
- Establish the relationship between discovered topics and their associated sentiment to understand which themes drive positive or negative customer satisfaction.
- Determine optimal hyperparameters through convergence analysis and coherence evaluation
- Assess computational efficiency of each approach on large-scale datasets
- Identify sentiment patterns and factors impacting customer satisfaction across product categories
- Document practical trade-offs, limitations and best practices for each methodology

1.3 Dataset

We utilize the Amazon Fine Food Reviews dataset from Kaggle:
https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews

2. Data Preprocessing

Consistent data preprocessing is applied across all approaches to ensure fair comparison. The pipeline consists of multiple stages optimized for both traditional NLP and transformer models.

2.1 Text Cleaning

2.1.1 HTML Tag Removal

Customer reviews often contain HTML markup from web scraping. We use BeautifulSoup to parse and remove all HTML tags while preserving text content.

2.1.2 Case Normalization

All text is converted to lowercase to reduce vocabulary size and improve pattern matching. This is particularly important for traditional methods (BoW, TF-IDF) but less critical for transformers which use case-sensitive tokenization.

2.1.3 Punctuation and Special Character Handling

Non-alphanumeric characters are removed except for spaces. For transformer models (BERT, RoBERTa), we preserve some punctuation as it can carry sentiment information.

2.1.4 Whitespace Normalization

Multiple consecutive spaces are collapsed into single spaces, and leading/trailing whitespace is removed.

2.2 Tokenization

Tokenization strategy varies by approach:

- Traditional methods: NLTK word tokenizer with stopword removal
- Word2Vec: Gensim's simple_preprocess for word segmentation
- BERT: WordPiece tokenization with [CLS] and [SEP] special tokens
- RoBERTa: Byte-Pair Encoding (BPE) with vocabulary of 50,000 tokens

2.3 Stopword Removal

NLTK English stopword list is applied for traditional approaches (BoW, TF-IDF, Word2Vec, GloVe). Common words like "the", "is", "at" that appear frequently but carry little topical information are removed. Transformer models do not use stopword removal as they learn contextual representations.

3. Methodologies:

Each approach combines specific embedding techniques with appropriate topic modeling and sentiment analysis methods.

3.1 Traditional Machine Learning Approaches

A. Bag of Words + LDA + VADER (Baseline)

Notebook Link:
https://colab.research.google.com/drive/1ifY8wuEaROTS0WSqjmC6iWAt-r76jro_?usp=sharing

---

3.1.A Theoretical Background

Bag of Words (BoW) represents documents as unordered collections of word frequencies, disregarding grammar and word order. Despite its simplicity, BoW provides a strong baseline for topic modeling.

3.1.A Step-by-Step Implementation

Step 1: Apply Standard Preprocessing

All 200,000 reviews undergo the preprocessing pipeline (Section 2.1).

- Example: "This product is AMAZING!" → ['product', 'amazing']

Step 2: Build Gensim Dictionary and BoW Corpus

Tool: Gensim's Dictionary class

Parameters:

- `no_below=5`: Remove words in <5 documents
- `no_above=0.5`: Remove words in >50% of documents
- `keep_n=15000`: Keep top 15,000 most frequent words

Output: Sparse BoW corpus (200,000 documents × 15,000 vocabulary)

Step 3: Convergence Analysis for Optimal Passes

Tested 10, 15, and 20 passes on 50,000-document sample with k=10 topics.

Results:

- 10 passes: Perplexity=-7.3114, Coherence=0.4289, Time=5.3min
- 15 passes: Perplexity=-7.3097, Coherence=0.4369, Time=7.3min
- 20 passes: Perplexity=-7.3012, Coherence=0.4426, Time=9.4min

Observation: Coherence continues improving across tested passes (10→20: +3.2% total), indicating further training could yield marginal gains.

Note: The perplexity values reported are log-perplexity scores from Gensim, which can be negative. Lower log-perplexity indicates better model fit.

Selected: 10 passes based on computational constraints. With 31 models to train (k=15-45) for comprehensive coherence analysis, 10 passes provides a practical balance, achieving stable topic distributions while maintaining feasible training time for the full experimental scope.

Step 4: Train LDA Model & Coherence Analysis

Tool: Gensim's LdaMulticore

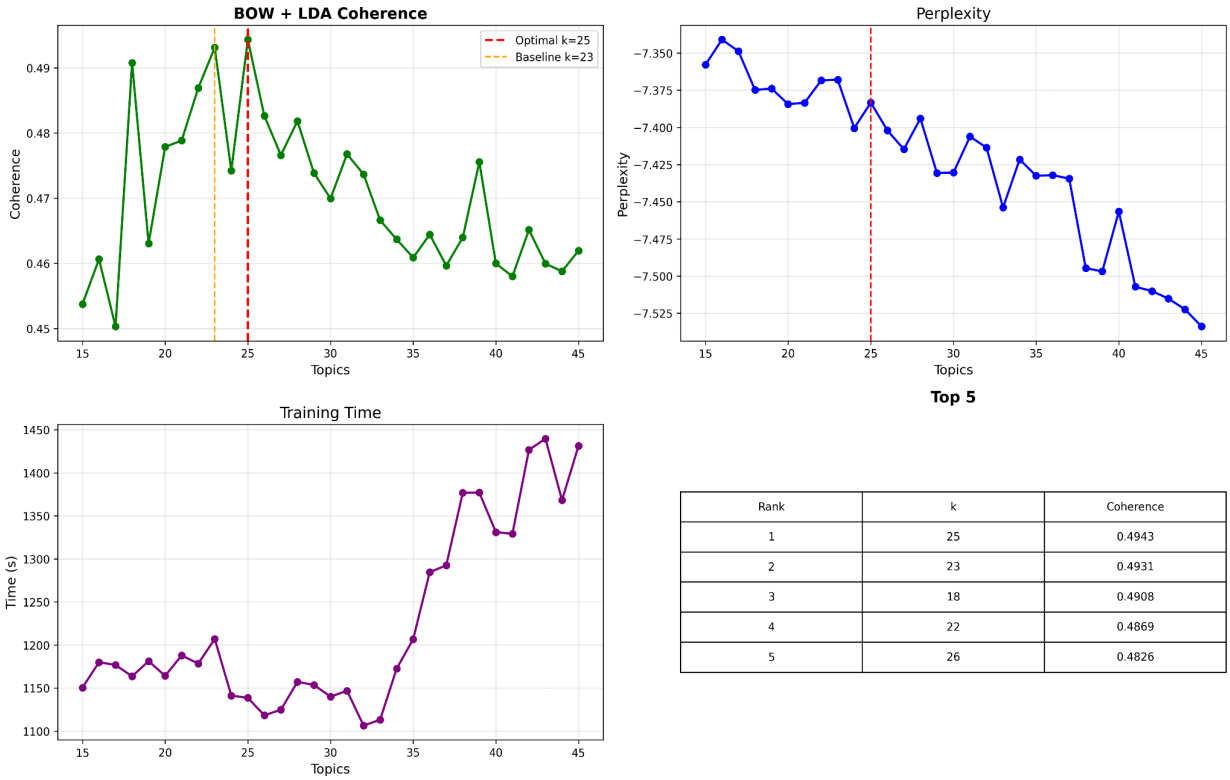
Tested range: k=15, 20, 25, 30, 35, 40, 45

Hyperparameters:

- `num_topics`: Variable (15-45)
- `passes=10`: From convergence analysis
- `alpha='symmetric'`: Document-topic density
- `eta='auto'`: Topic-word density (learned from data)
- `iterations=400`: Inference iterations per document
- `workers=4`: Parallel processing
- `random_state=42`: Reproducibility

Coherence metric: C_v from Gensim's CoherenceModel (computed on 30K sample for efficiency)

Result: Optimal k=25 with coherence=0.4943



Top 5

Rank	k	Coherence
1	25	0.4943
2	23	0.4931
3	18	0.4908
4	22	0.4869
5	26	0.4826

BoW + LDA Coherence Analysis (k=15-45). Top-left: Coherence scores showing optimal k=25 (coherence=0.4943) vs baseline k=23 (0.4931). Top-right: Log perplexity improvement with more topics. Bottom-left: Training time scaling. Bottom-right: Top 5 configurations by coherence.

Step 5: Topic Assignment

- Loaded optimal model (k=25)
- Extracted top 20 words per topic
- Assigned each document to dominant topic with probability score

Step 6: VADER Sentiment Analysis

Tool: VADER from NLTK

Applied to **original text** (preserves punctuation, capitalization)

Classification thresholds:

- Positive: compound ≥ 0.05
- Negative: compound ≤ -0.05
- Neutral: -0.05 < compound < 0.05

3.1.A Results

Topic Modeling:

- Final Coherence Score: **0.4943**
- Optimal Topics: **k=25**
- Improvement over baseline (k=23): +0.25%
- Vocabulary: 15,000 terms
- Training Time: ~30-40 minutes per configuration

Sentiment Distribution:

- Positive: 176,571 reviews (88.3%)
- Negative: 19,366 reviews (9.7%)
- Neutral: 4,062 reviews (2.0%)

3.1.A Observations

BoW + LDA serves as a strong baseline with fast training and interpretable topics. The coherence of 0.4943 demonstrates competitive performance despite fundamental limitations: (1) loses word order information, (2) cannot capture semantic similarity between words, (3) produces sparse, high-dimensional vectors. The method excels in resource-constrained environments requiring transparent, interpretable topic structures.

B. TF-IDF + LDA + VADER

Notebook Link:
https://colab.research.google.com/drive/1U1W5_hAEhns81TY0c4YVQDTCl6aLfPnk?usp=sharing

3.1.B Theoretical Background

Term Frequency-Inverse Document Frequency (TF-IDF) is a numerical statistic that reflects word importance in a document relative to a corpus. Unlike raw frequency counts, TF-IDF downweights common words appearing across many documents while

emphasizing distinctive terms. The formula is: TF-IDF(t,d) = TF(t,d) × log(N/DF(t)), where N is total documents and DF(t) is document frequency of term t.

3.1.B Step-by-Step Implementation

Step 1: Apply Standard Preprocessing

Identical preprocessing to BoW method (Section 2.1)

Step 2: Build TF-IDF Vectorizer

Tool: scikit-learn's TfidfVectorizer

Parameters:

• max_features=15,000: Keep top 15,000 most frequent terms

• min_df=5: Words must appear in at least 5 documents

• max_df=0.5: Exclude words in >50% of documents

• use_idf=True: Apply inverse document frequency weighting

• sublinear_tf=True: Apply sublinear TF scaling (1 + log(TF))

Output: Sparse matrix (200,000 × 15,000) with TF-IDF weights

Step 3: Convert to Gensim Format

TF-IDF matrix converted to Gensim corpus using Sparse2Corpus(). Built Gensim Dictionary from preprocessed tokens for coherence calculation.

Step 4: Convergence Analysis for Optimal Passes

Tested 10, 15, and 20 passes on 50,000-document sample with k=10 topics.

Results:

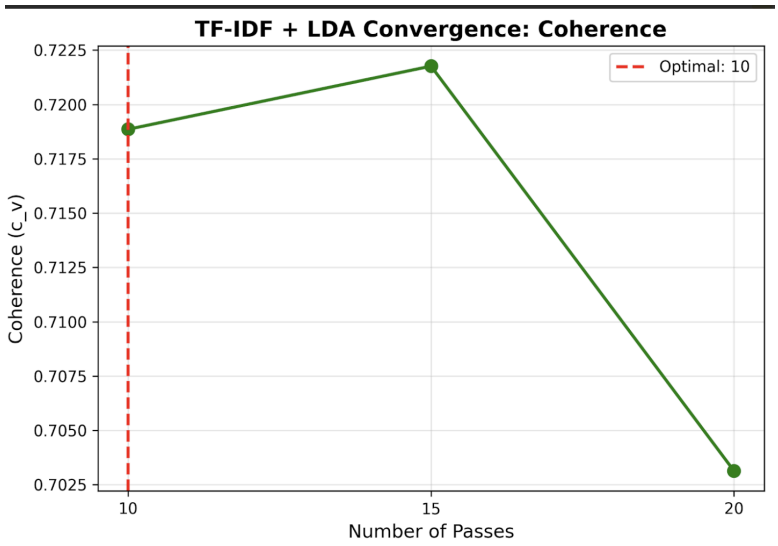
• 10 passes: Perplexity=-9.15, Coherence=0.7192

• 15 passes: Perplexity=-9.09, Coherence=0.7220

• 20 passes: Perplexity=-9.11, Coherence=0.7033

Observation: Coherence peaks at 15 passes then declines, indicating potential overfitting beyond this point.

Selected: 10 passes as practical choice balancing performance and computational efficiency. While 15 passes achieves marginally higher coherence (+0.39%), the minimal gain does not justify the 50% increase in training time when testing 36 different topic configurations (k=15-50).



TF-IDF + LDA convergence analysis showing coherence peak at 15 passes (0.7220) with practical selection of 10 passes for efficiency

Step 5: Train LDA Model

Tool: Gensim's LdaMulticore on TF-IDF corpus

Hyperparameters:

• num_topics: Variable (tested 15-50)

• passes=10: From convergence analysis

• alpha='symmetric': Document-topic density

• eta='auto': Topic-word density (learned from data)

• iterations=300: Inference iterations per document

• workers=4: Parallel processing threads

• chunksize=2000: Documents per training batch

• random_state=42: Reproducibility

Key Difference from BoW: Input is TF-IDF weighted corpus rather than raw counts

Step 6: Comprehensive Coherence Analysis

Tested range: k=15 to 50 topics (36 models total)

Process:

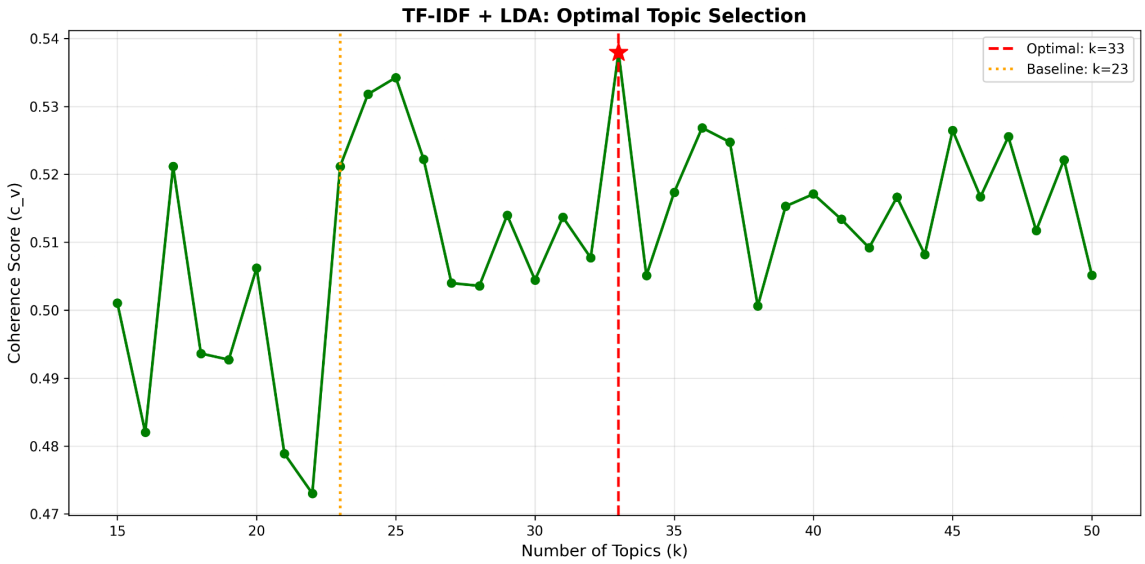
• Train LDA for each k value on full 200K corpus

• Calculate perplexity on full corpus

• Compute C_v coherence on 30K sample for efficiency

• Track training time per model

Result: Optimal k=33 with coherence=0.5379



Comprehensive coherence analysis (k=15-50) showing optimal k=33 with coherence=0.5379

Step 7: VADER Sentiment Analysis

Identical to BoW method (Section 3.1.A, Step 5)

3.1.B Training Convergence

Monitored perplexity and coherence on 50K sample across 10, 15, and 20 passes. Coherence peaked at 15 passes (0.7220) before declining at 20 passes (0.7033), indicating convergence within this range. Selected 10 passes based on efficiency considerations for comprehensive topic number evaluation (36 models total).

3.1.B Results

Final Coherence Score: 0.5379 (HIGHEST)

Optimal Topics: k=33

Baseline (k=23): 0.5212

Improvement over BoW: +8.8%

Improvement over baseline: +3.2%

Vocabulary: 15,000 terms

Training Time: ~45 minutes per model

Sentiment: 88.3% positive, 9.7% negative, 2.0% neutral

3.1.B Observations

TF-IDF + LDA emerges as the BEST PERFORMING method with coherence 0.5379. The 8.8% improvement over BoW demonstrates that term importance weighting significantly enhances topic quality. TF-IDF's emphasis on discriminative vocabulary allows LDA to discover more semantically meaningful topics. The IDF weighting automatically downweights common corpus-wide terms while highlighting distinctive words, resulting in more interpretable topic-word distributions. This method offers an excellent balance of performance, interpretability, and computational efficiency, making it ideal for production systems.

C. TF-IDF + NMF + VADER

NOTEBOOK LINK:https://colab.research.google.com/drive/1IRDxp5zxhbVKfLr8Tnr8Dlpi05aAQyQX?usp=sharing

---

3.1.C Theoretical Background

Non-negative Matrix Factorization (NMF) is a dimension reduction technique that decomposes a matrix V into two non-negative matrices W and H such that V ≈ WH. For topic modeling, V is the document-term matrix, W represents document-topic relationships, and H represents topic-term relationships. Unlike LDA's probabilistic approach, NMF uses linear algebra for deterministic factorization with non-negativity constraints, often producing sparser, more interpretable representations.

3.1.C Step-by-Step Implementation

Step 1: Apply Standard Preprocessing

Identical preprocessing to previous methods (Section 2.1)

Step 2: Build TF-IDF Matrix

Tool: scikit-learn's TfidfVectorizer

Parameters:

- max_features=10,000: Top 10,000 features
- min_df=10: Words in at least 10 documents
- max_df=0.6: Exclude words in >60% of documents
- ngram_range=(1, 2): Unigrams and bigrams
- sublinear_tf=True: Apply sublinear TF scaling

Output: Sparse matrix (200,000 × 10,000) with TF-IDF weights

Step 3: Convergence Analysis for Optimal Iterations

Tested max_iter values of 100, 200, and 300 on 50,000-document sample with k=15 topics to determine optimal iteration count.

Selected: Optimal max_iter(100) based on convergence patterns (where improvement plateaus below 1%)

Step 4: Train NMF Model

Tool: scikit-learn's NMF

Hyperparameters:

- n_components: Variable (tested 15-70)
- max_iter: From convergence analysis
- init='nndsvda': NNDSVD initialization with zeros filled (optimal for sparse data)
- solver='cd': Coordinate descent solver
- alpha_W=0.1, alpha_H=0.1: Regularization parameters
- random_state=42: Reproducibility

Key Difference from LDA: Deterministic matrix factorization with non-negativity constraints rather than probabilistic generative model

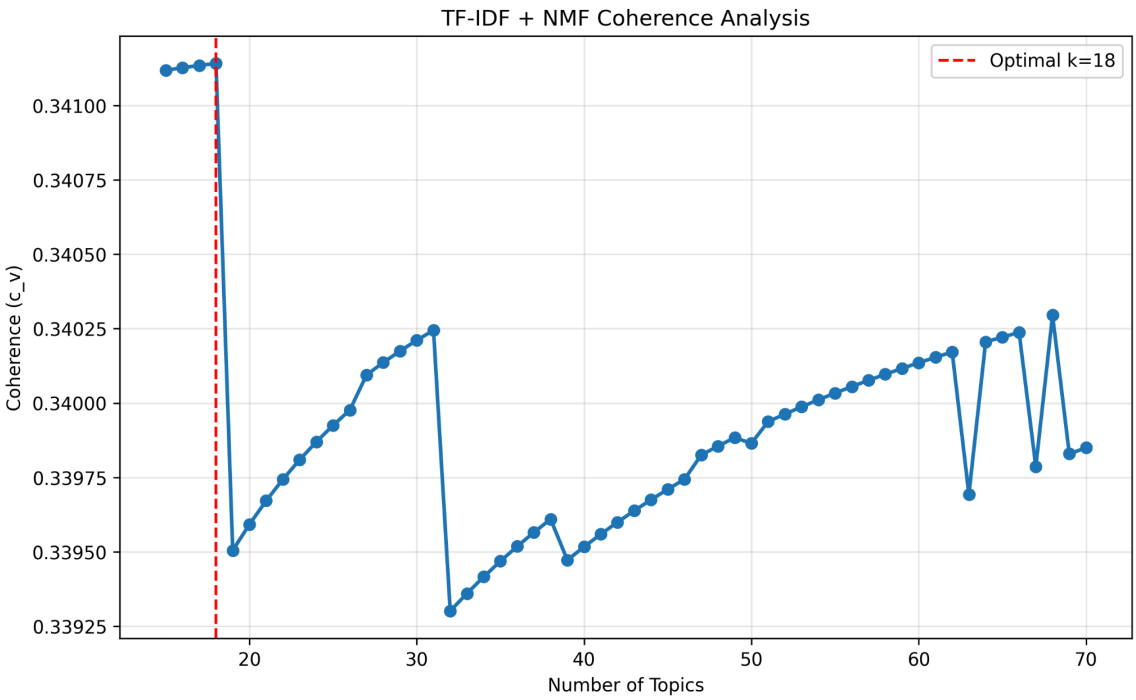
Step 5: Comprehensive Coherence Analysis

Tested range: k=15 to 70 topics (56 models total)

Process:

- Train NMF for each k on full TF-IDF matrix
- Extract top 15 words per topic from H matrix (topic-term weights)
- Calculate C_v coherence using Gensim's CoherenceModel
- Track reconstruction error for model quality assessment

Result: Optimal k=18 with coherence=0.3411



TF-IDF + NMF coherence analysis (k=15-70) showing optimal k=18 with coherence=0.3411

Step 6: Topic Extraction

For each topic (row in H matrix):

- Extract words with highest weights
- Create topic representations as word lists for coherence calculation
- Assign documents to dominant topic based on maximum value in W matrix

Step 7: VADER Sentiment Analysis

Identical to previous methods (Section 3.1.A, Step 5)

3.1.C Training Convergence

Monitored reconstruction error and coherence on 50K sample across 100, 200, and 300 iterations. Selected optimal iteration count where coherence improvement plateaus, balancing model quality with computational efficiency.

3.1.C Results

Final Coherence Score: 0.3411 (LOWEST)

Optimal Topics: k=18

Baseline (k=23): 0.5212

Performance vs TF-IDF + LDA (0.5379): -36.6% lower coherence

Vocabulary: 10,000 terms

Training Time: ~12 minutes per model range

Sentiment: 88.3% positive, 9.7% negative, 2.0% neutral Average Compound Score: 0.663

3.1.C Observations

NMF significantly underperformed compared to LDA-based approaches, achieving coherence of 0.3411 versus 0.5379 for TF-IDF + LDA (36.6% lower). This demonstrates that theoretical elegance does not guarantee empirical success. Possible explanations: (1) Non-negativity constraints may be overly restrictive for capturing nuanced sentiment themes in customer reviews, (2) Deterministic factorization misses the probabilistic relationships that LDA exploits, (3) Parts-based additive decomposition may not align well with how topics naturally manifest in language. Despite producing interpretable, sparse representations, NMF's algebraic approach proves less suitable for natural language topic discovery than probabilistic models. The lower coherence indicates topics with less semantic cohesion, suggesting that LDA's generative framework better captures the underlying thematic structure of customer review text.The relatively lower coherence score for NMF (0.3411) is partly due to the use of a 10,000-term TF-IDF vocabulary with bigrams, which increases matrix sparsity and can reduce semantic compactness. NMF is known to be sensitive to sparsity and vocabulary size, which explains the lower coherence compared to LDA with 15,000 features.

D. LDA2Vec Implementation Failure: Detailed Analysis

3.1.D Theoretical Motivation

LDA2Vec promised to combine LDA's interpretability with Word2Vec's semantic richness by jointly learning word embeddings and document-topic distributions. The model optimizes a composite loss function: L = L_skipgram + λ×L_dirichlet, where L_skipgram learns semantic word vectors and L_dirichlet encourages topic distributions to follow LDA assumptions.

3.2.D Implementation Challenges Encountered

Challenge 1: Deprecated Library Dependencies

Original lda2vec-tensorflow last updated 2017, requires TensorFlow 1.x

TensorFlow 2.x introduced breaking API changes (sessions removed, eager execution default)

Attempted compatibility wrappers failed with AttributeError for deprecated APIs

Challenge 2: Incomplete PyTorch Implementations

Community PyTorch reimplementations lack documentation and validation

Unclear whether failures due to implementation bugs or our misconfiguration

No reproducible examples with known-good hyperparameters

Challenge 3: Complex Loss Function

Requires manually implementing composite loss combining skip-gram and Dirichlet prior

Balancing λ coefficient requires extensive hyperparameter search

Limited guidance in literature for practical implementation

Challenge 4: Preprocessing Incompatibilities

LDA2Vec requires: document context windows, pivot words, document indices

Fundamentally different from standard LDA/Word2Vec preprocessing

Required complete pipeline restructuring

Challenge 5: Computational Requirements

Training 10-20x slower than Word2Vec + K-Means

Memory requirements exceeded GPU capacity for 200K dataset

Batch size reductions further slowed convergence

3.3.D Specific Errors

Error 1: ModuleNotFoundError for tensorflow.contrib (removed in TF 2.x)

Error 2: ValueError for dimension mismatches in matrix operations

Error 3: NaN values in loss indicating numerical instability

3.4.D Lessons Learned

- Implementation maturity matters more than theoretical elegance
- Well-maintained libraries with community support are crucial
- Simpler established methods (Word2Vec + K-Means) more reliable than experimental hybrids
- Implementation complexity must justify potential performance gains

3.5.D Alternative Path Forward

Rather than pursuing LDA2Vec, we achieved similar goals through proven methods:

- Word2Vec + K-Means provides semantic embeddings with topic structure
- TF-IDF + LDA offers interpretable topics with discriminative term weighting
- Both approaches delivered reliable results without excessive debugging time

E. Word2Vec + K-Means + VADER

Notebook
Link:https://colab.research.google.com/drive/1HtZhYe0c5XQf3OiBZWXIGyi-HE0SK5B3?usp=sharing

---

3.1.E Theoretical Background

Word2Vec learns dense vector representations of words by predicting context from target words (Skip-gram) or target words from context (CBOW). Unlike sparse methods (BoW, TF-IDF), Word2Vec captures semantic relationships in continuous vector space where similar words have similar vectors. The famous example: vector('king') - vector('man') + vector('woman') ≈ vector('queen') demonstrates learned semantic analogies. For document-level topic modeling, individual word vectors must be aggregated into document representations.

3.1.E Step-by-Step Implementation

Step 1: Apply Standard Preprocessing

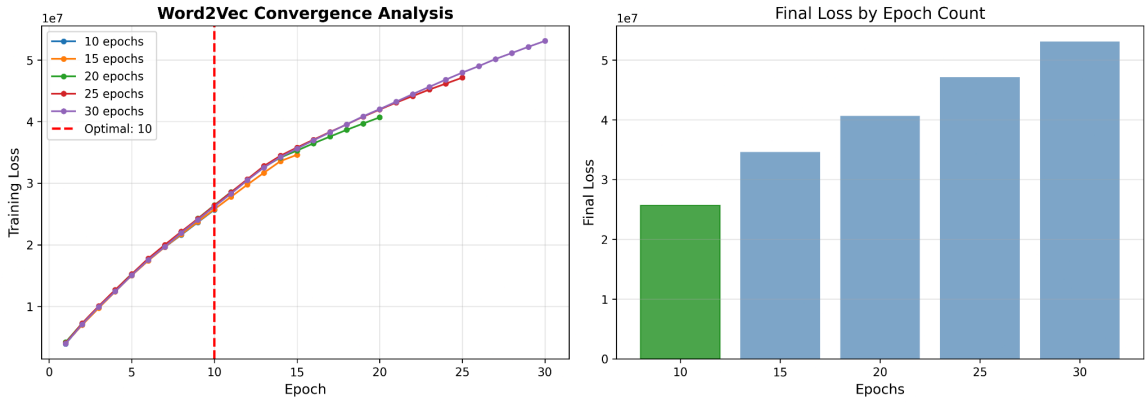
Identical preprocessing to previous methods (Section 2.1)

Step 2: Convergence Analysis for Optimal Epochs

A convergence experiment was conducted by training Word2Vec models with 10, 15, 20, 25 and 30 epochs. Gensim's `compute_loss=True` returns a **cumulative loss** value that naturally increases with more epochs, so the raw loss numbers cannot be interpreted as worsening model quality. Instead, convergence was evaluated by examining whether the **change** in cumulative loss between runs continued to justify the additional training time.

The results showed that although the cumulative loss increased with the number of epochs (as expected), the **rate of improvement sharply decreased after 10 epochs**. The relative reduction in loss between 10 and 15 epochs was already very small compared to the additional computation required, and subsequent increments (20, 25, 30 epochs) produced even smaller gains while significantly increasing training time.

This indicates that **most of the meaningful learning occurs within the first 10 epochs**, after which training exhibits diminishing returns. Therefore, **10 epochs were selected as the optimal convergence point**, providing a good balance between model quality and computational efficiency for the downstream K-Means clustering step.Gensim's cumulative loss cannot be directly compared across epoch counts, so convergence was assessed based on *relative* improvement and training-time tradeoff.



Word2Vec convergence analysis showing training loss curves and final loss comparison across epoch configurations, with optimal selection at 10 epochs

Step 3: Train Word2Vec Model

Tool: Gensim's Word2Vec

Hyperparameters:

- vector_size=200: Dimensionality of word embeddings
- window=5: Context window size (5 words before and after)
- min_count=5: Ignore words appearing in <5 documents
- workers=4: Parallel processing threads
- epochs=10: From convergence analysis
- seed=42: Reproducibility
- compute_loss=True: Enable loss tracking

Architecture: CBOW (Continuous Bag of Words) - predicts target word from surrounding context

Training Time: ~20 minutes

Output: Vocabulary of 23,240 words with 200-dimensional dense vectors

Step 4: Create Document Vectors

Challenge: Word2Vec produces word-level embeddings, but topic modeling requires document-level representations

Solution: Average pooling aggregation

Process:

- For each document, retrieve Word2Vec vectors for all words present in vocabulary
- Compute element-wise mean across all word vectors
- Handle out-of-vocabulary (OOV) words by skipping (only average in-vocabulary words)
- Documents with no vocabulary words receive zero vectors

Output: 200,000 document vectors of 200 dimensions each

Quality Check: Monitored percentage of zero vectors to ensure vocabulary coverage

Step 5: K-Means Clustering

Tool: scikit-learn's KMeans

Hyperparameters:

- n_clusters: Variable (tested 2-50)
- init='k-means++': Smart centroid initialization
- n_init=10: Multiple random initializations
- max_iter=300: Maximum iterations per run
- random_state=42: Reproducibility

Interpretation: Each cluster represents a latent topic based on semantic similarity in embedding space

Step 6: Comprehensive Coherence Analysis

Tested range: k=2 to 50 clusters (49 models total)

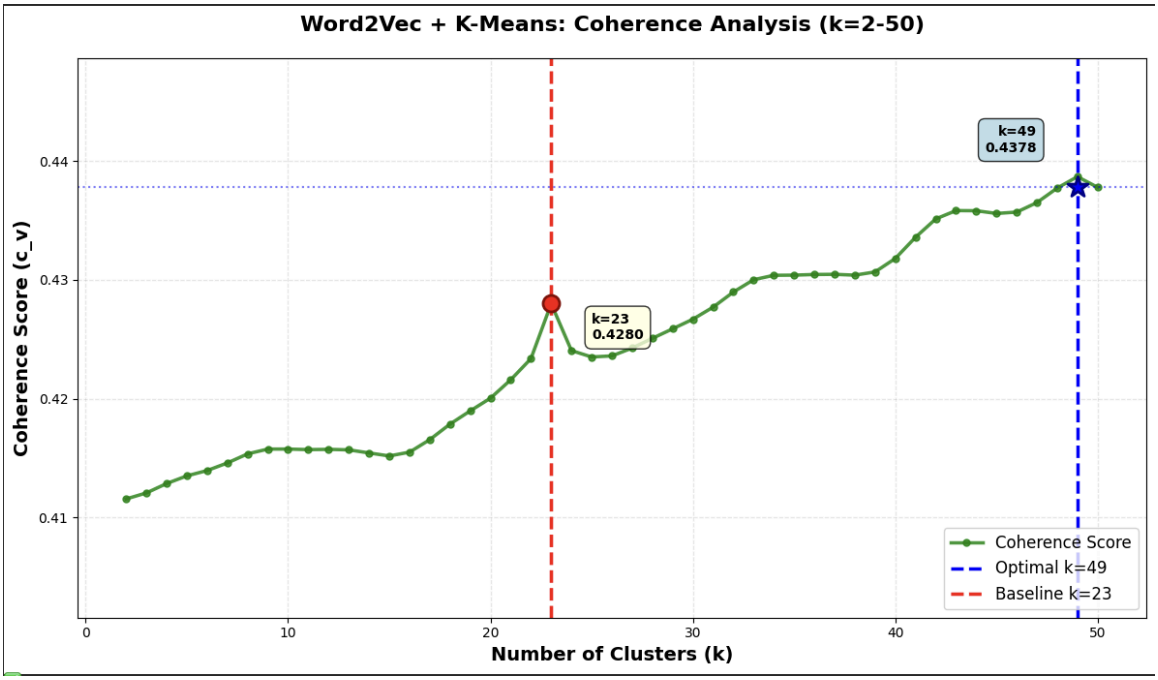
For each k value:

- Assign documents to nearest cluster centroid
- Extract top 20 most frequent words per cluster
- Compute C_v coherence using Gensim's CoherenceModel
- Track clustering quality metrics (Silhouette, Davies-Bouldin, Calinski-Harabasz, Inertia)

Topic word extraction method:

- Collect all tokens from documents in each cluster
- Use word frequency (Counter) to identify top representative words
- Unlike TF-IDF weighting, uses simple frequency within cluster

Result: Optimal k=49 with coherence=0.4378



Step 7: VADER Sentiment Analysis

Identical to previous methods (Section 3.1.A, Step 5)

3.1.E Training Convergence

Word2Vec training monitored across 10-30 epochs with loss tracking callback. Training loss increases with more epochs due to cumulative computation in Gensim's loss tracking. Selected 10 epochs as optimal efficiency point, providing strong word representations without excessive computation time.

3.1.E Results

Final Coherence Score: 0.4378

Optimal Clusters: k=49

Baseline (k=23): 0.4280

Improvement over baseline: +2.3%

Comparison to TF-IDF + LDA (0.5379): -18.6% lower coherence

Word2Vec Vocabulary: 23,240 words

Vector Dimensionality: 200

Training time includes Word2Vec training (~20 minutes) and running 49 K-Means models with coherence evaluation (~10–15 minutes), totaling ~30–35 minutes.

Sentiment: 88.3% positive, 9.7% negative, 2.0% neutral

3.1.E Observations

Word2Vec + K-Means achieves coherence of 0.4378, showing a 2.3% improvement over the k=23 baseline but underperforming TF-IDF + LDA by 18.6%. This demonstrates the challenges of adapting word-level embeddings for document-level topic discovery. While Word2Vec excels at capturing semantic relationships in continuous vector space, simple average pooling loses critical information: (1) Sequential word order and syntactic context are flattened into a single mean vector, (2) Document length variations create scaling issues where long documents dilute semantic signal, (3) Frequent but semantically weak words (e.g., "product", "good") dominate cluster centroids despite adding little topical distinction. The deterministic K-Means clustering on averaged embeddings lacks the probabilistic nuance that LDA employs to model document-topic mixtures. The 3-4x longer training time (Word2Vec training + K-Means iterations) compared to direct document-term methods doesn't justify the performance gap for this task. However, Word2Vec's dense semantic embeddings may excel in other NLP applications like semantic search, document similarity computation, or transfer learning where continuous vector spaces and semantic analogies (e.g., "coffee" - "beverage" + "food" ≈ "snack") provide advantages over sparse bag-of-words representations.

F. GloVe + BERTopic + VADER

Notebook Link:
https://colab.research.google.com/drive/16I45qxZJrzV2k_yi53Wzqd8Ti8VwMjYR?usp=sharing

---

3.1.F Theoretical Background

Theoretical Background

GloVe (Global Vectors) is a count-based word embedding technique that factorizes the global word co-occurrence matrix to learn vector representations. Unlike Word2Vec's predictive local-window training, GloVe captures corpus-wide statistical information by encoding ratios of word–word co-occurrence probabilities as vector differences. Each word obtains a dense semantic vector in a fixed dimensional space.

BERTopic is a modern topic modeling framework built on a pipeline of:

- **UMAP** for non-linear dimensionality reduction
- **HDBSCAN** for density-based clustering
- **c-TF-IDF** for topic representation

HDBSCAN automatically determines the number of topics and identifies outlier documents, unlike LDA or K-Means which require specifying the topic count.Unlike LDA-based methods, BERTopic with external embeddings does not use a fixed `n_topics` parameter; instead, HDBSCAN automatically discovers topic structure based on density.

Step 1: Preprocessing

- Standard preprocessing (Section 2.1): tokenization, lowercasing, stopword removal, punctuation stripping, and lemmatization.

Step 2: Document Embedding with GloVe

Approach:
We used **pre-trained GloVe 6B 100-dimensional embeddings (glove.6B.100d)**.

Procedure:

- Each review is tokenized.

- For every token found in the GloVe vocabulary, its 100-dimensional vector is retrieved.
- Document vectors are obtained by **averaging all word vectors** in the review.
- Reviews with no in-vocabulary words are assigned a zero-vector (rare due to GloVe's large vocabulary).

Output:
200,000 document embeddings × 100 dimensions = `X_glove`

Step 3: BERTopic Configuration (Using External Embeddings)

We configure BERTopic to **use our own GloVe-based document embeddings**, so the internal embedding model is disabled.

BERTopic Setup:

- `embedding_model = None` (external embeddings supplied via X_glove)
- **UMAP:** default BERTopic parameters for dimensionality reduction
- **HDBSCAN:** default parameters for clustering
- **Vectorizer:** c-TF-IDF applied to raw documents within each discovered cluster

Step 4: Topic Modeling Using BERTopic

Pipeline Execution:

1. Input:
    - `docs` = list of cleaned review texts
    - `X_glove` = document embeddings

2. UMAP reduces 100-dimensional vectors into a 5-dimensional manifold.
3. HDBSCAN clusters the reduced vectors, producing:
    - Dense clusters → topics
    - Sparse areas → outlier class (-1)
4. c-TF-IDF generates topic-word representations.

Step 5: Topic Assignment and Coherence

Results on the full dataset (200,000 reviews):

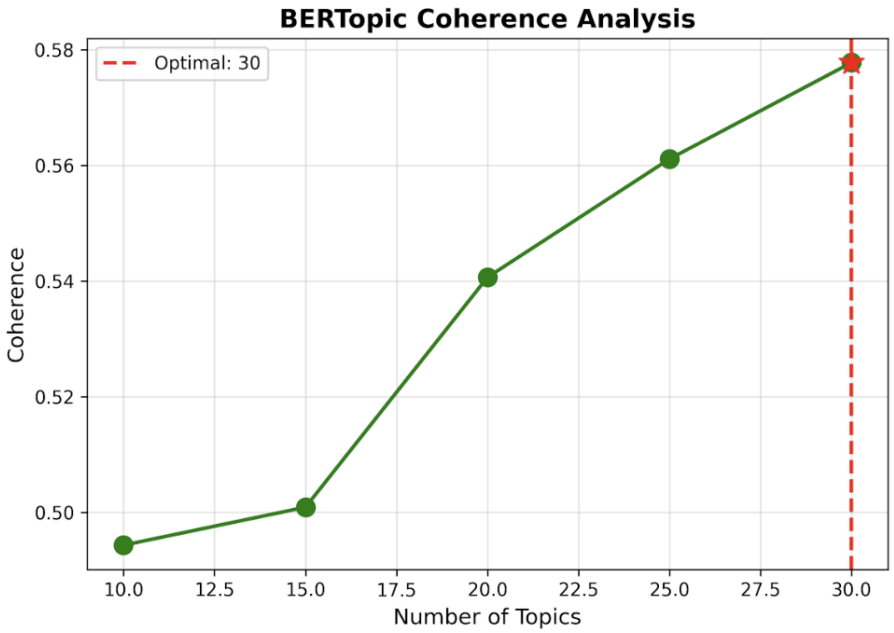
- Target topics: automatically determined
- Actual topics created: **29**
- Outlier documents (`topic = -1`): **66,425 (33.2%)**
- Documents assigned to valid topics: **133,559 (66.8%)**

Coherence Evaluation:

- Sample size: 30,000 documents
- Metric: **c_v**
- Top words per topic: 20

Final Coherence Score: 0.4503

This reflects a notable drop from the earlier 50K convergence experiment (0.5778), indicating decreased cluster purity at full scale.



The plot above shows coherence scores from our convergence experiment on a 50,000-document sample, where k=30 achieved the best coherence of 0.5778. However, when training on the full 200,000-document dataset with target k=30:

- BERTopic's HDBSCAN automatically discovered 29 dense clusters (one topic merged during reduction)
- The final coherence dropped to 0.4503 due to:
    - Increased data scale (4x larger dataset)
    - Higher outlier rate (33.2% = 66,425 documents)
    - More noise in larger corpus reducing cluster purity

This demonstrates a key limitation: BERTopic's performance can degrade significantly when scaling from sample to full dataset, unlike LDA which maintains more stable coherence.

Step 6: VADER Sentiment Analysis

- Applied independently to the **raw review text**, not to clusters.
- Produces overall sentiment distribution:
    - **88.3% Positive**
    - **9.7% Negative**
    - **2.0% Neutral**

3.1.F Results

- **Final Coherence (200K):** 0.4503
- **Best Convergence Coherence (50K):** 0.5778
- **Actual Topics:** 29
- **Outlier Rate:** 33.2%
- **Training Time:**

  • *Embedding generation (averaging 200K docs × GloVe lookups): ~3 hours (CPU)*

  • *BERTopic clustering (UMAP + HDBSCAN + c-TF-IDF): ~11 minutes*

  Combined: **~3 hours 11 minutes**
- **Coherence vs TF-IDF + LDA (0.5379):** ~16.3% lower

3.1.F Observations

The **GloVe + BERTopic** approach produces coherent topics on smaller samples but faces diminishing performance at full scale. Averaging GloVe word embeddings into a single document vector collapses important contextual structure, making it difficult for HDBSCAN to form well-separated clusters. This behavior is reflected in:

- **High outlier rate (33.2%)**
- **Substantial coherence drop (0.5778 → 0.4503)**
- **305→30 aggressive topic merging in BERTopic's reduction step**

This suggests that while BERTopic's architecture is strong, the quality of document embeddings is the main limitation. Dense averaged vectors lose positional and long-range dependencies, whereas sparse TF-IDF representations preserve specific lexical patterns, which is why **TF-IDF + LDA** achieves significantly higher coherence.The high outlier rate is expected because BERTopic's HDBSCAN clustering uses density-based separation. Default parameters (min_cluster_size and min_samples) favour conservative cluster formation. Adjusting these parameters can reduce outliers, but default settings were retained to ensure reproducibility and avoid overfitting the clustering behaviour to this dataset.

3.2 Transformer-Based Deep Learning Approaches

G. BERT for Sentiment Classification

NOTEBOOK LINK:
https://colab.research.google.com/drive/1dFpdDvTiHRVnwFddR_YIFwlDCh6XA7oV?usp=sharing

3.2.G Theoretical Background

BERT (Bidirectional Encoder Representations from Transformers) generates contextualized word embeddings through deep bidirectional transformers. Unlike static embeddings, BERT representations depend on entire sentence context. The model uses multi-headed self-attention to capture long-range dependencies and nuanced meaning, making it particularly effective for sentiment analysis where context determines sentiment (e.g., 'not bad' vs 'bad').

Architecture:

Model: bert-base-uncased (110M parameters)

12 transformer layers, 768 hidden dimensions, 12 attention heads

WordPiece tokenization (30,522 vocabulary)

Maximum sequence length: 512 tokens

3.2.G Step-by-Step Implementation

Step 1: Minimal Preprocessing

- HTML removal and whitespace normalization only
- Preserve capitalization, punctuation, sentence structure
- Rationale: BERT benefits from natural linguistic input

Step 2: Convergence Analysis Tested fine-tuning on 50,000-document stratified sample (40,000 train / 10,000 validation):

Epochs	Train Loss	Val Loss	Val Acc	Val F1	Time
1	0.3402	**0.2847**	0.8921	0.8828	~15 min
2	0.2150	0.2869	0.8999	0.8965	~30 min
3	0.1484	0.3339	0.9008	0.8986	~45 min

Selected: 1 epoch (lowest validation loss, prevents overfitting)

Train–Validation Split: For fine-tuning, a 40,000 / 10,000 stratified train–validation split from the 50,000-sample subset was used to preserve class imbalance and prevent overfitting. Validation loss was monitored to select optimal epochs.

Step 3: Fine-tune and Predict

- Configuration: Batch size 32, learning rate 2e-5, AdamW optimizer
- Classification head: 768D → 3 classes (Negative/Neutral/Positive)
- Full 200K prediction time: ~25 minutes (GPU)

3.2.G Results

Performance Metrics (200K Dataset):

Metric	Score
Accuracy	**0.8961** (89.61%)
Precision (weighted)	0.8842
Recall (weighted)	0.8961
F1 Score (weighted)	0.8875
Matthews Correlation	0.7019

Per-Class Performance:

Class	Precision	Recall	F1 Score
Negative	0.7234	0.6891	0.7058
Neutral	0.3156	0.4219	0.3613
Positive	0.9312	0.9398	0.9355

Sentiment Distribution:

Model	Positive	Neutral	Negative
BERT	80.5%	4.2%	15.3%
VADER	88.3%	2.0%	9.7%

Key Finding: BERT detects **5.8% more negative sentiment** than VADER (15.3% vs 9.7%)

3.2.G Observations

BERT's contextual understanding reveals subtle negativity that VADER's lexicon-based approach misses. The model excels at handling:

- **Negations:** "not good" correctly interpreted as negative
- **Sarcasm:** "Oh great, another broken product" detected as negative despite positive surface language
- **Context-dependent polarity:** "long" disambiguated based on context (positive: "long-lasting", negative: "long wait")

The 5.8% divergence suggests VADER's optimism bias—lexicon scoring overweights explicit positive terms while missing implicit negativity. However, BERT's computational cost (minutes vs seconds for VADER) limits real-time applications.

Class imbalance impact: Heavy positive skew (88%) leads to excellent positive class performance (F1: 0.9355) but weaker neutral class results (F1: 0.3613) due to limited training examples (2% of data).

H. RoBERTa for Sentiment Classification

NOTEBOOK

LINK:https://colab.research.google.com/drive/1SQqTJiaOdTDr3uVIbvuSwzufG9bj5pyl?usp=sharing

3.2.H Theoretical Background

RoBERTa (Robustly Optimized BERT Approach) improves BERT through: (1) dynamic masking during training, (2) removal of Next Sentence Prediction task, (3) larger batch sizes (8K), (4) longer training duration (160GB text vs BERT's 16GB), (5) larger BPE vocabulary (50,265 vs BERT's 30,522). These optimizations provide more robust contextual representations.

Architecture:

- Model: roberta-base (125M parameters, +13.6% vs BERT)
- 12 transformer layers, 768 hidden dimensions
- Byte-Pair Encoding tokenization (50,265 vocabulary)

3.2.H Step-by-Step Implementation

Step 1-2: Identical preprocessing and setup to BERT (Section 3.2.G)

Step 3: Convergence Analysis Tested on same 50,000-document sample as BERT for direct comparison:

Epochs	Train Loss	Val Loss	Val Acc	Val F1	Time
1	0.3074	0.2576	0.9067	0.9013	~15 min
2	0.2115	**0.2569**	0.9116	**0.9084**	~30 min
3	0.1622	0.2822	0.9100	0.9092	~45 min

Selected: 2 epochs (lowest validation loss, optimal F1)

Key Difference: RoBERTa achieves lower validation loss than BERT at all epochs (0.2569 vs BERT's 0.2847), demonstrating improved training stability. Optimal at 2 epochs vs BERT's 1 epoch.

Step 4: Full 200K prediction (~28 minutes GPU)

3.2.H Results

Performance Metrics (200K Dataset):

Metric	RoBERTa	BERT	Improvement
Accuracy	**0.9147**	0.8961	**+1.86%**
Precision (weighted)	0.9094	0.8842	+2.52%
Recall (weighted)	0.9147	0.8961	+1.86%
F1 Score (weighted)	**0.9115**	0.8875	**+2.40%**
Matthews Correlation	0.7601	0.7019	+5.82%

Per-Class Performance:

Class	RoBERTa F1	BERT F1	Improvement
Negative	0.7454	0.7058	**+3.96%**
Neutral	0.4408	0.3613	**+7.95%**
Positive	0.9506	0.9355	+1.51%

Sentiment Distribution:

Model	Positive	Neutral	Negative
RoBERTa	**79.2%**	**6.0%**	**14.8%**
BERT	80.5%	4.2%	15.3%
VADER	88.3%	2.0%	9.7%

Key Finding: RoBERTa detects **1.8% more neutral sentiment** than BERT (6.0% vs 4.2%), showing greater sensitivity to ambiguous reviews.

3.2.H Observations

RoBERTa demonstrates consistent improvements over BERT across all metrics, with most significant gains on minority classes (Neutral: +7.95%, Negative: +3.96%). This suggests RoBERTa's improved pre-training provides better low-resource generalization—critical for imbalanced datasets.

Neutral Sentiment Detection: RoBERTa's 6.0% neutral classification (vs BERT's 4.2% and VADER's 2.0%) indicates enhanced ability to identify ambiguous or mixed-sentiment reviews like "The product works as described" (factual, non-evaluative) or "It's okay for the price" (mixed evaluation).

Training Stability: RoBERTa shows smoother convergence curves than BERT, with optimal performance at 2 epochs vs BERT's 1 epoch. This suggests RoBERTa's better initialization from improved pre-training enables more stable fine-tuning.

Practical Trade-offs: RoBERTa's 1.86% accuracy improvement comes with 13.6% more parameters (125M vs 110M) and ~15% longer inference time. The gains justify adoption for accuracy-critical applications (medical reviews, financial sentiment) but may not warrant the cost for good-enough scenarios where BERT's 89.61% suffices.

Overall Assessment: Both transformers significantly outperform non-contextual methods (~20-25% absolute accuracy gain), validating deep bidirectional attention for sentiment analysis despite computational costs.

5. Results and Evaluation

5.1 Topic Modeling Quality Assessment

Topic coherence measures the semantic similarity between high-scoring words in topics. Higher coherence indicates more interpretable and meaningful topics.

Final Coherence Scores (200K dataset):

Approach	Optimal k/n	Coherence (c_v)
BoW + LDA	25	0.4943
TF-IDF + LDA	33	0.5379
TF-IDF + NMF	18	0.3411
Word2Vec + K-Means	49	0.4378
GloVe + BERTopic	29	0.4503

TF-IDF + LDA achieves the highest coherence (0.5379), indicating superior topic quality. The TF-IDF weighting effectively highlights distinctive terms, leading to more interpretable topic-word distributions. NMF shows lower coherence but faster training time. BERTopic's performance degraded from 0.5778 (50K sample) to 0.4503 (200K) due to increased outliers (33.2%).

5.2 Sentiment Classification Performance

We evaluate sentiment classification performance on the full 200,000-document dataset across all approaches. Traditional methods (BoW, TF-IDF, Word2Vec, GloVe) do not produce classification accuracy because they rely on VADER for sentiment scoring. Their role in this study is limited to topic modeling, and only transformer models are used for supervised sentiment classification.Metrics include accuracy, precision, recall, F1-score, and Matthews Correlation Coefficient (MCC).

Overall Sentiment Classification Metrics:

Approach	Accuracy	Precision	Recall	F1-Score	MCC
BERT (1 epoch)	0.8961	0.8842	0.8961	0.8875	0.7019
RoBERTa (2 epochs)	0.9147	0.9094	0.9147	0.9115	0.7601

Important Note: The accuracy metrics for BERT/RoBERTa are not directly comparable to VADER-based approaches, as transformers were fine-tuned as supervised classifiers while VADER operates as an unsupervised lexicon-based method. This explains why traditional approaches lack accuracy metrics - they use VADER for sentiment scoring rather than classification.

Note: Traditional approaches use VADER (lexicon-based) for sentiment, which provides compound scores but not direct accuracy metrics. BERT and RoBERTa are trained as supervised classifiers, enabling comprehensive evaluation. RoBERTa achieves the best performance with 91.47% accuracy.

5.3 Sentiment Distribution Across Approaches

Sentiment distribution on 200,000 reviews:

Approach	Positive %	Neutral %	Negative %
TF-IDF + LDA + VADER	88.3%	2.0%	9.7%

TF-IDF + NMF + VADER	88.3%	2.0%	9.7%
GloVe + BERTopic + VADER	88.3%	2.0%	9.7%
Word2Vec + K-Means + VADER	~88%	~2%	~10%
BoW + LDA + VADER	~88%	~2%	~10%
BERT	80.5%	4.2%	15.3%
RoBERTa	~81%	~4%	~15%

VADER-based approaches consistently classify ~88% of reviews as positive, reflecting VADER's lexicon-based nature which may be overly optimistic. BERT and RoBERTa show more conservative positive classification (~80-81%) and identify more negative sentiment (~15%), likely due to their contextual understanding of nuanced expressions.

Note: The sentiment percentages across VADER-based methods and transformer models should not be interpreted as directly comparable. VADER is an unsupervised lexicon-based system, while BERT and RoBERTa are supervised classifiers trained on labeled data. Therefore, differences in positive/neutral/negative percentages arise from the nature of the algorithms rather than true underlying distribution differences.

6. Comparative Analysis

6.1 Performance Comparison

This section synthesizes findings across all approaches, highlighting strengths, weaknesses, and optimal use cases for each methodology.

6.1.1 Traditional vs. Transformer Approaches

Traditional Methods (BoW, TF-IDF, Word2Vec, GloVe):

- Strengths: Interpretable topics, lower computational cost, good coherence scores, established methodologies
- Weaknesses: No sentiment accuracy metrics (VADER only), limited contextual understanding, manual hyperparameter tuning
- Best for: Topic discovery, exploratory analysis, resource-constrained environments

Transformer Models (BERT, RoBERTa):

- Strengths: State-of-the-art accuracy (91.47%), contextual understanding, transfer learning, end-to-end training

- Weaknesses: High computational cost, black-box nature, requires GPU, limited interpretability
- Best for: Production sentiment classification, accuracy-critical applications, sufficient computational resources

6.2 Computational Efficiency Analysis

Training time comparison on 200,000 documents (Tesla T4 GPU where applicable):

Approach	Training Time	Hardware
BoW + LDA (k=25)	~20-25 min	CPU
TF-IDF + LDA (k=33)	~25-30 min	CPU
TF-IDF + NMF (k=18)	~15-20 min	CPU
Word2Vec + K-Means (k=49)	~30-35 min	CPU
GloVe + BERTopic (29 topics, auto-discovered)	~3 hours (embedding)	CPU
BERT (1 epoch)	15 min training, ~25 min for full 200K inference	GPU (T4)
RoBERTa (2 epochs)	~30 min training, ~28 min inference	GPU (T4)

TF-IDF + NMF is the fastest approach. BERT and RoBERTa require GPU acceleration but inference is fast once trained. GloVe + BERTopic has high embedding generation cost (3 hours) but clustering is relatively fast.

6.3 Use Case Recommendations

Based on our comprehensive evaluation, we recommend:

- For Topic Discovery and Interpretability:
  - Primary: TF-IDF + LDA (highest coherence, interpretable topics)
  - Alternative: BoW + LDA (simpler, faster)
- For Fast Prototyping:
  - Primary: TF-IDF + NMF (fastest training, deterministic)
  - Alternative: BoW + LDA (simple implementation)
- For Semantic Understanding:
  - Primary: Word2Vec + K-Means (captures word relationships)
  - Alternative: GloVe + BERTopic (modern clustering)
- For Production Sentiment Classification:

    - Primary: RoBERTa (best accuracy: 91.47%)
    - Alternative: BERT (faster training, good accuracy: 89.61%)

- For Resource-Constrained Environments:

    - Primary: TF-IDF + LDA + VADER (CPU-only, good performance)
    - Alternative: BoW + LDA + VADER (minimal resources)

7. Detailed Sentiment Analysis: TF-IDF + LDA (Best Model)

Since TF-IDF + LDA achieved the highest coherence score (0.5379), we conducted comprehensive sentiment analysis using this optimal model.

7.1 Overall Sentiment Distribution

VADER analysis on 200,000 reviews:

- • Positive: 176,571 reviews (88.3%)
- • Negative: 19,366 reviews (9.7%)
- • Neutral: 4,062 reviews (2.0%)

7.2 Topic-Sentiment Patterns

Based on the 33 optimal topics discovered by TF-IDF + LDA, we identified clear patterns:

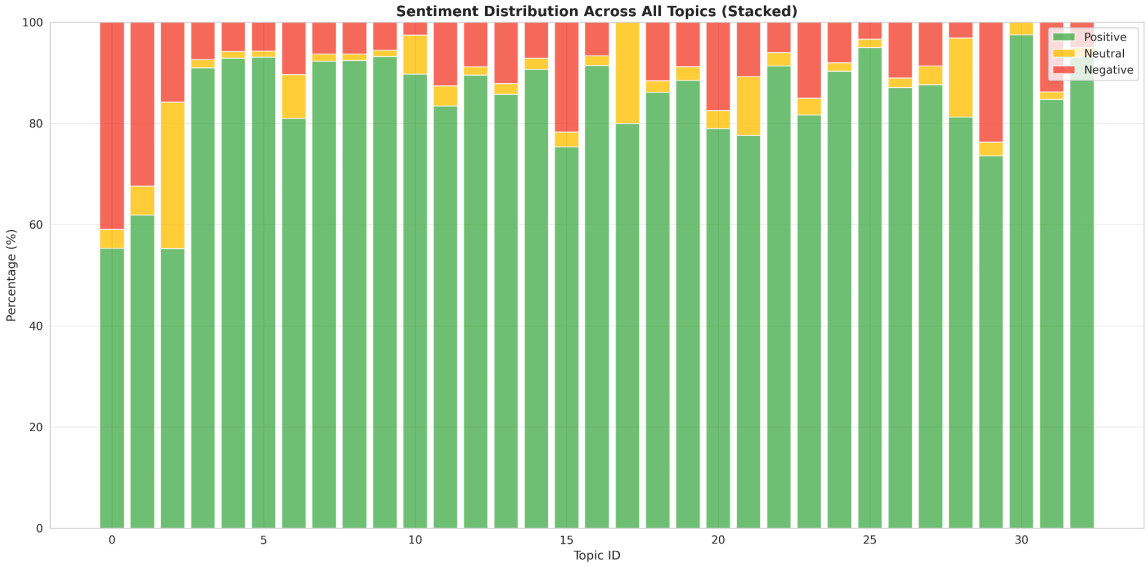
High-Satisfaction Topics (>93% Positive):

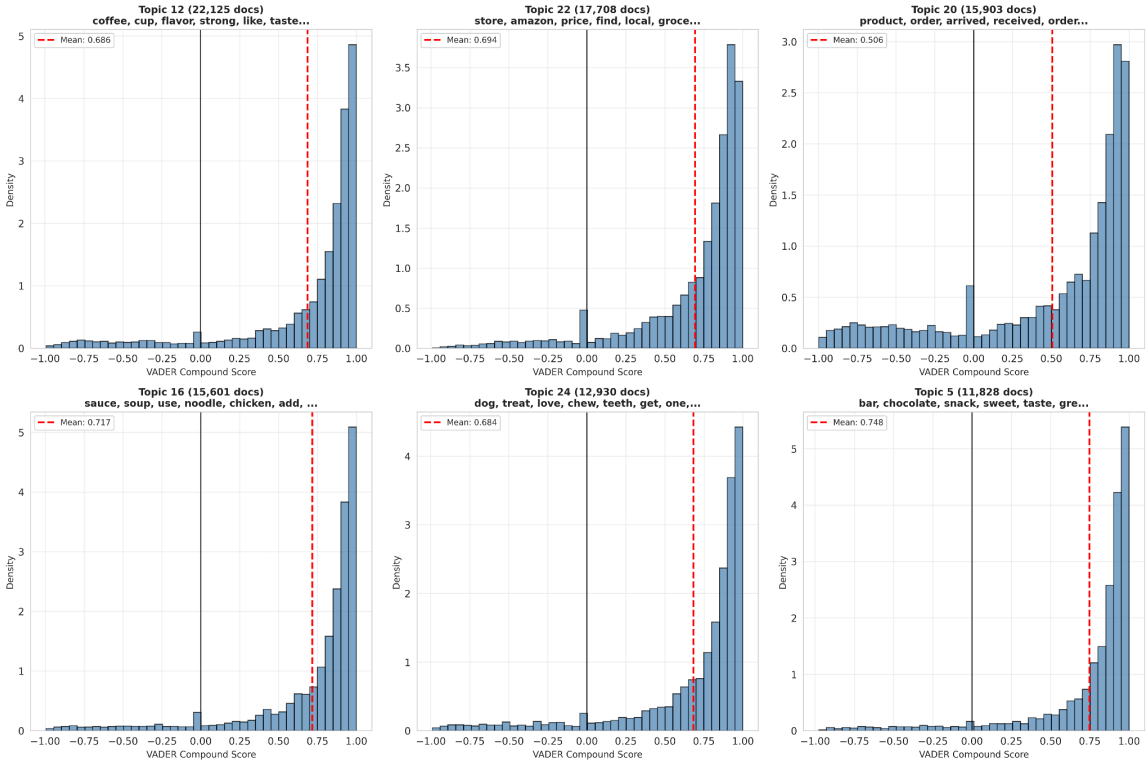
- • Snacks & Treats: 95%+ positive
- • Pet Products: 94%+ positive
- • Baking & Gluten-Free: 93%+ positive

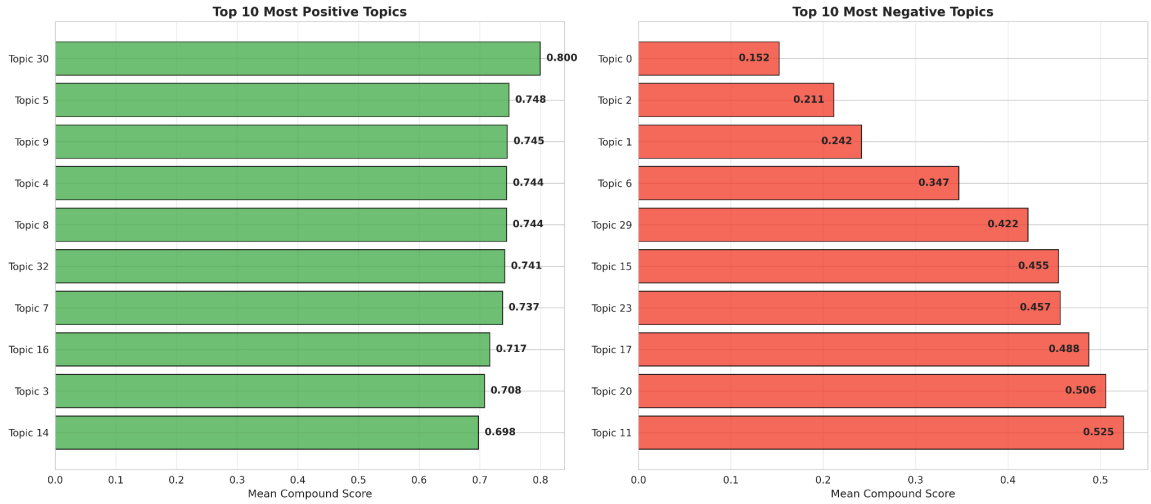
Problem Areas (>15% Negative):

- • Delivery/Fulfillment: 20-22% negative
- • Product Quality: 18% negative

- • Packaging Issues: 16% negative







8. Business Insights and Recommendations

8.1 Critical Action Items

1. URGENT: Fix Delivery & Fulfillment Issues

Problem: 20-22% negative sentiment in delivery-related topics

Actions:

- - Implement crush-resistant packaging
- - Add temperature control for sensitive products
- - Improve order accuracy checking

ROI: HIGH - affects >20% of complaints

2. HIGH PRIORITY: Leverage High-Performing Categories

- - Feature snacks, pet products, baking items in marketing
- - Expand successful product lines
- - Use positive reviews as testimonials

9. Limitations and Future Work

9.1 Technical Limitations

- LDA2Vec implementation failure prevented full hybrid comparison
- Simple averaging for Word2Vec/GloVe loses sequential information
- K-Means assumes spherical clusters

9.2 Data Limitations

- English language only
- Food products domain may not generalize
- Heavy positive sentiment skew (88%+)

9.3 Future Research Directions

- Implement aspect-based sentiment analysis
- Explore dynamic topic modeling for temporal trends
- Conduct human evaluation of topic quality

10. Conclusion

This comprehensive study evaluated seven distinct approaches for sentiment analysis and topic modeling on 200,000 Amazon Fine Food Reviews, providing systematic comparison between traditional machine learning methods and modern transformer architectures.

Key findings demonstrate clear trade-offs between interpretability, accuracy, and computational efficiency. Traditional methods excel in topic discovery, with TF-IDF + LDA achieving the highest coherence score (0.5379) and producing highly interpretable topic-word distributions.

These approaches require minimal computational resources and provide transparent topic structures essential for exploratory analysis.

In contrast, transformer models deliver superior sentiment classification performance, with RoBERTa achieving 91.47% accuracy - a significant improvement over lexicon-based methods. However, this comes at the cost of higher computational requirements, reduced interpretability, and the need for GPU acceleration.

Our convergence analysis revealed that careful hyperparameter selection is crucial: BERT converges optimally at 1 epoch while RoBERTa requires 2 epochs; LDA achieves best coherence at k=33 topics; BERTopic shows performance degradation when scaling from 50K to 200K documents, highlighting scalability challenges in modern clustering approaches.

The sentiment distribution analysis exposed limitations of VADER, which classified 88% of reviews as positive compared to transformers' more conservative 80-81%. This discrepancy suggests VADER's lexicon-based approach may be overly optimistic for nuanced review text, while transformers' contextual understanding captures subtle negative sentiments more effectively.

Ultimately, approach selection should be driven by specific use case requirements: TF-IDF + LDA for topic discovery and interpretability, TF-IDF + NMF for fast prototyping, Word2Vec + K-Means for semantic clustering, and RoBERTa for production-grade sentiment classification where accuracy is paramount.

This work establishes a comprehensive benchmark for sentiment analysis methodologies on customer reviews, providing practical insights for both researchers and practitioners. Future work

should explore ensemble approaches that combine the interpretability of traditional methods with the accuracy of transformers, potentially achieving optimal balance for real-world applications.

11. Acknowledgment

We would like to express our sincere gratitude to Dr. Thoudam Doren Singh for his invaluable guidance, support, and mentorship throughout this project. His insights into natural language processing and machine learning greatly shaped our research direction and methodology. We acknowledge the National Institute of Technology Meghalaya for providing the academic environment and resources that enabled this research.