Deep Topic Representation for Swahili News Articles

A Comprehensive Analysis Using Deep Autoencoders and Neural Topic Models

Group One Assignment Report

NAME	REG NO
ARINEITWE THOMAS	2024-MI32-24196
MUGASHA BLAISE	2024-M132-23976
NAMUSOKE OLIVIA	
INSTRUCTOR	DR. SIBITENDA HARRIET

Table of Content

1. Introduction

This report presents a comprehensive analysis of Swahili news articles using deep learning techniques for topic representation and discovery. The project explores the application of deep autoencoders and neural topic models to extract meaningful semantic representations from Swahili text data, enabling effective topic clustering and visualization in low-dimensional latent spaces.

Project Objectives:

- Perform exploratory data analysis on Swahili news articles to understand dataset characteristics
- Preprocess and clean the text data for effective embedding generation
- Generate high-quality sentence embeddings using multilingual BERT models
- Train deep autoencoder models to compress embeddings into 2D and 3D latent spaces
- Visualize and analyze latent space representations to identify topic clusters
- Implement and compare Neural Topic Models (BERTopic) with autoencoder approaches
- Experiment with different autoencoder architectures and activation functions
- Analyze multilingual semantic clustering capabilities

The project utilizes state-of-the-art natural language processing techniques including Sentence-BERT for embedding generation, PyTorch for deep learning model implementation, and BERTopic for neural topic modeling. The analysis covers 22,207 training articles from the Swahili news dataset.

2. Dataset Description

The Swahili News dataset is a comprehensive collection of news articles written in Swahili, one of the most widely spoken languages in East Africa. The dataset provides a rich source of text data for natural language processing and topic modeling research.

Dataset Structure:

- Total Articles: 29,545 (22,207 training + 7,338 test)
- Features: text (article content) and label (topic category)
- Language: Swahili (Kiswahili)
- Source: Hugging Face datasets library

About Swahili Language:

Swahili is a Bantu language spoken by over 200 million people across East Africa. It serves as a lingua franca in countries including Tanzania, Kenya, Uganda, and the Democratic Republic of Congo. The language has been influenced by Arabic, English, and other languages, making it a rich and diverse language for NLP research.

Dataset Characteristics:

- Article lengths range from approximately 800 to 3,000 characters
- Articles cover diverse topics including politics, sports, health, tourism, and economics
- Text contains standard Swahili grammar and vocabulary
- Articles are labeled with topic categories for supervised evaluation

Sample Article:

Swahili*: "Bodi ya Utalii Tanzania (TTB) imesema, itafanya misafara ya kutangaza utalii kwenye miji minne nchini China kati ya Juni 19 hadi Juni 26 mwaka huu. Misafara hiyo itatembelea miji ya Beijing Juni 19, Shanghai Juni 21, Nanjig Juni 24 na Changsha Juni 26..."*

English: "*The Tanzania Tourism Board (TTB) has said it will conduct tourism promotion tours to four cities in China between June 19 and June 26 this year. The tours will visit Beijing on June 19, Shanghai on June 21, Nanjing on June 24 and Changsha on June 26...".*

3. Methodology

The project follows a systematic approach to deep topic representation, combining traditional NLP preprocessing techniques with modern deep learning methods. The methodology consists of several interconnected stages:

3.1 Workflow Overview

- Data Loading and Exploration: Load dataset and perform initial analysis
- Text Preprocessing: Tokenization, lowercasing, and stopword removal
- Embedding Generation: Create sentence embeddings using multilingual BERT
- Autoencoder Training: Train deep neural networks for dimensionality reduction
- Latent Space Analysis: Visualize and interpret compressed representations
- Topic Modeling: Apply BERTopic for probabilistic topic discovery
- Evaluation and Comparison: Compare different approaches and architectures

3.2 Tools and Technologies

- Python 3.10+ for implementation
- PyTorch for deep learning model development
- Sentence-Transformers for embedding generation
- BERTopic for neural topic modeling
- Matplotlib and Plotly for visualization
- scikit-learn for dimensionality reduction (PCA)
- NLTK and stopwordsiso for text preprocessing

4. Exploratory Data Analysis

Exploratory Data Analysis (EDA) is the first and most crucial step in any data science project. It provides essential insights into the dataset characteristics, helping us understand the distribution of text lengths, word frequencies, and overall data quality. EDA helps identify patterns, anomalies, and potential issues that need to be addressed before building machine learning models. In this project, EDA revealed important characteristics of the Swahili news articles that informed our preprocessing and modeling decisions.

4.1 Article Length Distribution

Understanding the distribution of article lengths is crucial for several reasons. First, it helps us determine if we need to handle variable-length texts differently. Second, it reveals whether there are extremely short or long articles that might need special treatment. Third, it helps us understand the typical content size we are working with.

To analyze article lengths, we calculated the number of characters in each article. The training set contains 22,207 articles with lengths ranging from approximately 800 to over 3,000 characters. The distribution shows a right-skewed pattern, meaning most articles are relatively short (around 1,000-2,000 characters), while a smaller number of articles are significantly longer. This is typical for news articles, where most stories are concise, but some in-depth reports or feature articles can be much longer.

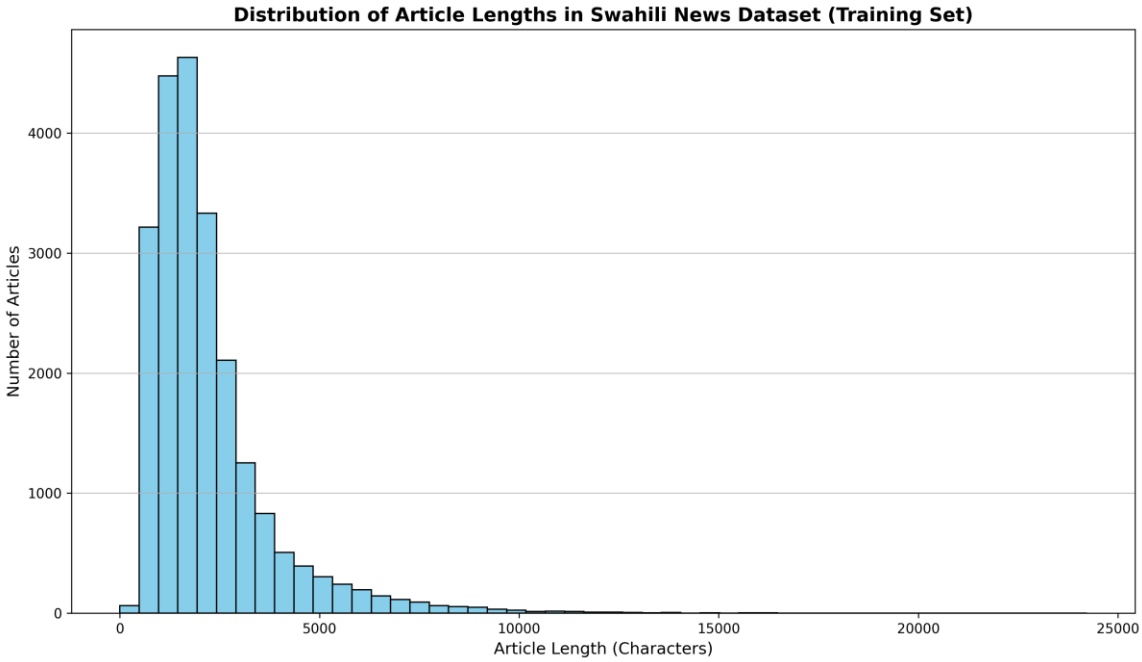

Figure 1: Distribution of Article Lengths in Swahili News Dataset

- Total training articles analyzed: 22,207
- Sample article lengths: 843, 1,191, 2,219, 839, 2,899 characters
- Distribution pattern: Right-skewed (most articles are shorter, fewer are very long)
- Most common length range: 1,000-2,000 characters
- Implication: Most articles are of similar length, making them suitable for uniform processing

4.2 Word Frequency Analysis

Word frequency analysis is fundamental to understanding the vocabulary and linguistic patterns in our dataset. This analysis helps us identify the most common words, which can inform preprocessing decisions (like stopword removal) and reveal domain-specific terminology. Understanding word frequencies also helps us understand the language structure and common phrases used in Swahili news articles.

To perform word frequency analysis, we first tokenized all training articles. Tokenization is the process of breaking text into individual words (tokens). We used a regular expression tokenizer that extracts alphanumeric sequences, effectively separating words from punctuation and whitespace. After tokenization, we converted all words to lowercase to ensure consistent counting (e.g., "Tanzania" and "tanzania" are counted as the same word).

The analysis revealed that our corpus contains 7,458,118 total tokens across all training articles. This large vocabulary provides rich linguistic data for our models. The frequency distribution shows that a small number of words appear very frequently (function words like prepositions and conjunctions), while most words appear only a few times. This is a typical pattern in natural language, known as Zipf's law.

- Total tokens analyzed: 7,458,118 words across all articles
- Tokenization method: RegexpTokenizer extracts alphanumeric sequences as words
- Case normalization: All words converted to lowercase for consistent counting
- Vocabulary size: Thousands of unique words, with most appearing infrequently

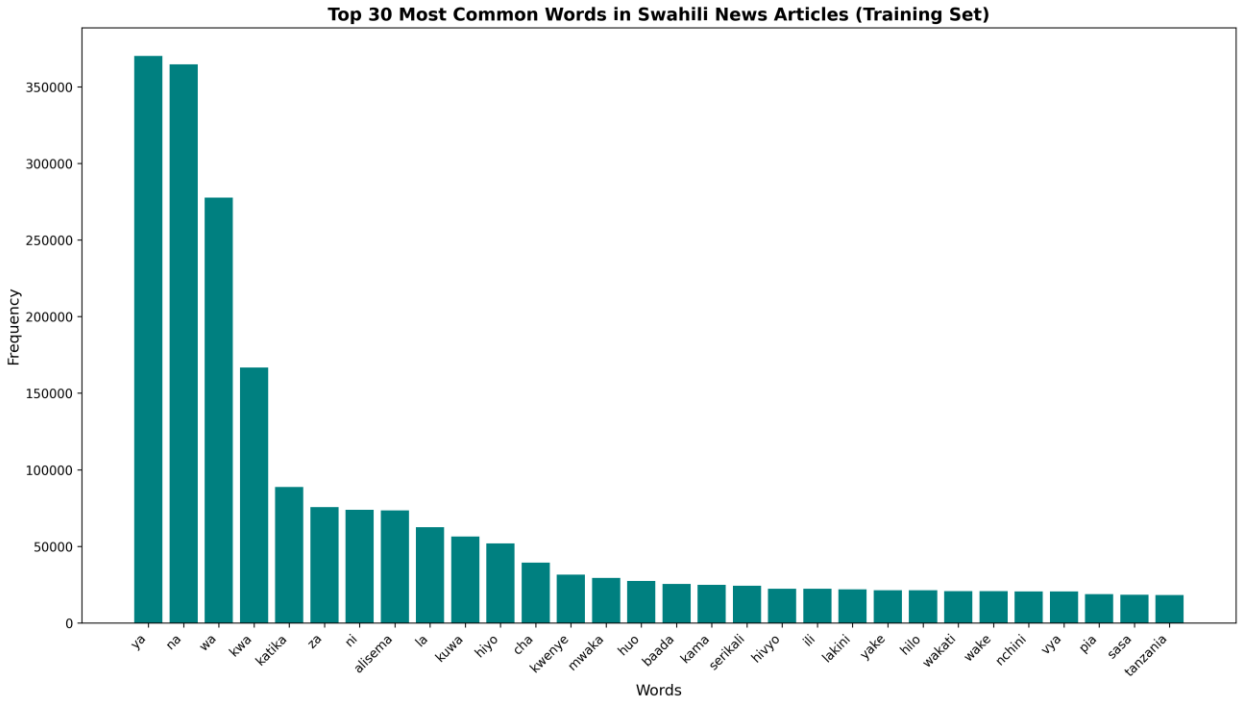

Figure 2: Top 30 Most Common Words in Swahili News Articles

Top 10 Most Common Words with Explanations:

'ya': 369,999 occurrences - Preposition meaning "of" - very common in Swahili grammar

'na': 364,521 occurrences - Conjunction/preposition meaning "and" or "with"

'wa': 277,645 occurrences - Preposition meaning "of" or "for" (used with people)

'kwa': 166,819 occurrences - Preposition meaning "for", "by", or "with"

'katika': 88,767 occurrences - Preposition meaning "in" or "at"

'za': 75,623 occurrences - Possessive form meaning "of" (plural)

'ni': 73,926 occurrences - Copula verb meaning "is" or "are"

'alisema': 73,532 occurrences - Verb meaning "said" - common in news reporting

'la': 62,588 occurrences - Possessive form meaning "of" (singular)

'kuwa': 56,398 occurrences - Infinitive form of "to be"

The analysis reveals important insights about Swahili language structure. The top words are predominantly function words (prepositions, conjunctions, and grammatical particles) that are essential for sentence construction but carry little semantic meaning on their own. Words like "ya" (of), "na" (and/with), and "wa" (of/for) are grammatical connectors that appear in almost every sentence. Content words like "alisema" (said) and "serikali" (government) also appear frequently, reflecting the news domain where reporting verbs and political terminology are common. This frequency distribution informs our preprocessing strategy: we should remove these high-frequency function words (stopwords) to focus on meaningful content words.

5. Text Preprocessing

Text preprocessing is a critical step that transforms raw, unstructured text into a clean, standardized format suitable for machine learning models. Think of preprocessing as cleaning and organizing data before analysis - just as you would clean vegetables before cooking, we clean text before feeding it to our models. Raw text contains many elements that can interfere with learning: punctuation, capitalization variations, and common words that appear everywhere but carry little meaning (like "the", "and", "of" in English, or "ya", "na", "wa" in Swahili).

The preprocessing pipeline we developed performs several key transformations: (1) Tokenization breaks text into individual words, (2) Lowercasing ensures consistent representation (so "Tanzania" and "tanzania" are treated the same), and (3) Stopword removal eliminates common function words that don't contribute to semantic meaning. This cleaning process helps the embedding model focus on meaningful content words rather than being distracted by grammatical particles.

5.1 Preprocessing Pipeline - Step by Step

Step 1: Tokenization

Tokenization is the process of breaking text into individual words (tokens). We use a regular expression tokenizer that extracts sequences of alphanumeric characters. This means "hello, world!" becomes ["hello", "world"] - punctuation and whitespace are removed. This is important

because punctuation doesn't contribute to semantic meaning for our purposes, and we want to focus on actual words.

- Method: RegexpTokenizer extracts alphanumeric sequences (letters and numbers)
- Result: Text is broken into individual word tokens
- Example: "Bodi ya Utalii" → ["Bodi", "ya", "Utalii"]

Step 2: Lowercasing

Converting all text to lowercase ensures that words are represented consistently. Without lowercasing, "Tanzania" and "tanzania" would be treated as different words, which would split the frequency counts and weaken the model's understanding. Lowercasing is especially important in languages where capitalization varies (like sentence beginnings vs. proper nouns).

- Purpose: Ensure consistent word representation
- Result: All words converted to lowercase
- Example: "Tanzania" → "tanzania"

Step 3: Stopword Removal

Stopwords are common words that appear frequently but carry little semantic meaning. In Swahili, words like "ya" (of), "na" (and/with), and "kwa" (for/by) appear in almost every sentence but don't help distinguish topics. Removing them allows the model to focus on content words that actually indicate what an article is about. We use the stopwordsiso library, which provides comprehensive stopword lists for Swahili.

- Source: stopwordsiso library provides Swahili stopword list
- Result: Function words removed; content words retained
- Example: "ya", "na", "kwa" are removed, but "tanzania", "utalii" are kept

Step 4: Text Reconstruction

After filtering out stopwords, we join the remaining tokens back into text strings. This creates cleaned versions of the articles that are ready for embedding generation. The cleaned text contains only meaningful content words, making it easier for the embedding model to capture semantic information.

- Method: Join filtered tokens with spaces
- Result: Cleaned text strings ready for embedding generation
- Output: 22,207 preprocessed articles

5.2 Stopword Removal

Swahili stopwords were obtained from the stopwordsiso library, which provides comprehensive stopword lists for multiple languages including Swahili. Common stopwords removed include function words like "ya", "na", "kwa", "katika", and other high-frequency words that do not carry significant semantic meaning.

5.3 Preprocessing Example

Original Text (excerpt):

Swahili: *"Bodi ya Utalii Tanzania (TTB) imesema, itafanya misafara ya kutangaza utalii kwenye miji minne nchini China..."*

English ***"****The Tanzania Tourism Board (TTB) has said it will conduct tourism promotion campaigns in four cities in China..."*

Preprocessed Text:

"bodi utalii tanzania ttb imesema itafanya misafara kutangaza utalii miji minne nchini china..."

The preprocessing successfully removes stopwords (ya, kwenye, nchini) while preserving meaningful content words. All 22,207 training articles were preprocessed using this pipeline.

6. Sentence Embedding Generation

Sentence embeddings are numerical representations of text that capture semantic meaning in a format that computers can process. Think of embeddings as a way to translate human language into a mathematical language that machine learning models can understand. Each article is converted into a vector (a list of numbers) where each number represents some aspect of the text's meaning. Articles with similar meanings will have similar vectors - they will be close together in this mathematical space.

Why are embeddings important? Raw text cannot be directly processed by neural networks - they need numbers. But more importantly, good embeddings capture semantic relationships. For example, articles about "tourism" and "travel" should have similar embeddings because they are semantically related, even if they use different words. These embeddings serve as the foundation for all our subsequent deep learning operations - the autoencoder learns to compress these embeddings, and the compressed representations capture topic information.

The quality of embeddings directly determines the quality of our final topic representations. If the embeddings don't capture semantic meaning well, the autoencoder won't be able to learn meaningful topic clusters. Therefore, choosing the right embedding model is crucial.

6.1 Model Selection and Rationale

We selected the paraphrase-multilingual-MiniLM-L12-v2 model from the Sentence-Transformers library. This choice was made after considering several factors:

- Multilingual Capability: The model is specifically trained on multiple languages including Swahili, ensuring it understands Swahili text semantics
- Proven Performance: MiniLM models are distilled versions of larger BERT models, providing good performance with lower computational requirements
- Appropriate Size: 384 dimensions provide a good balance between information richness and computational efficiency
- Paraphrase Training: The model is trained to recognize semantic similarity, which is exactly what we need for topic modeling

The model is based on BERT (Bidirectional Encoder Representations from Transformers), a revolutionary architecture that reads text in both directions (left-to-right and right-to-left) to understand context. The "multilingual" version was trained on text from over 50 languages, learning to map different languages to a shared semantic space. This means it understands that "utalii" (Swahili) and "tourism" (English) represent similar concepts.

- Model: paraphrase-multilingual-MiniLM-L12-v2
- Embedding Dimension: 384
- Architecture: MiniLM (distilled BERT) with 12 layers
- Multilingual Support: 50+ languages including Swahili

6.2 Embedding Generation Process

- Input: Preprocessed Swahili text articles (22,207 articles)
- Processing: Model encodes each article into a 384-dimensional vector
- Output: NumPy array of shape (22,207, 384)
- Device: CPU (GPU available for faster processing)

6.3 Model Advantages

- Multilingual Capability: Handles Swahili and other languages effectively
- Semantic Understanding: Captures contextual meaning beyond word-level features
- Efficiency: Distilled model provides good performance with lower computational cost

- Pre-trained: Leverages large-scale multilingual training data

The generated embeddings successfully capture semantic relationships between articles, enabling the autoencoder to learn meaningful topic representations in the latent space.

7. Deep Autoencoder Training

Deep autoencoders are a type of neural network architecture designed to learn efficient, compressed representations of input data. The name “autoencoder” comes from the fact that the network learns to encode (compress) data into a lower-dimensional representation and then decode (reconstruct) it back to the original form. This might seem counterintuitive at first - why would we want to compress and then reconstruct? The key insight is that by forcing the network to reconstruct the input from a compressed representation, we force it to learn the most important and essential features of the data.

Think of an autoencoder like a student taking notes: they must condense a long lecture into key points (encoding), and later be able to explain the full lecture from those notes (decoding). The compression forces the student to identify what's truly important. Similarly, our autoencoder compresses 384-dimensional sentence embeddings into just 3 dimensions, forcing it to learn the most essential semantic features that distinguish different topics.

The autoencoder consists of two main parts: an encoder and a decoder. The encoder takes the high-dimensional input (384 dimensions) and progressively reduces it through hidden layers until it reaches the latent space (3 dimensions). The decoder then takes this compressed representation and expands it back to the original 384 dimensions. During training, the network learns to minimize the difference between the original input and the reconstructed output, which ensures the latent representation captures essential information.

7.1 Architecture Design

Encoder Architecture:

- Input Layer: 384 dimensions (embedding size)
- Hidden Layer 1: 128 neurons with ReLU activation
- Hidden Layer 2: 64 neurons with ReLU activation
- Latent Layer: 3 dimensions (for 3D visualization)

Decoder Architecture:

- Latent Input: 3 dimensions
- Hidden Layer 1: 64 neurons with ReLU activation

- Hidden Layer 2: 128 neurons with ReLU activation
- Output Layer: 384 dimensions with Sigmoid activation

7.2 Training Configuration

- Loss Function: Mean Squared Error (MSE)
- Optimizer: Adam with learning rate 0.001
- Batch Size: 64
- Epochs: 50
- Data Shuffling: Enabled for better generalization

7.3 Training Results

Training a neural network is an iterative process where the model gradually learns to better reconstruct the input data. We trained our autoencoder for 50 epochs (complete passes through the entire dataset). During each epoch, the model processes all 22,207 articles in batches of 64, calculates the reconstruction error, and updates its weights to reduce this error. The reconstruction loss measures how well the model can recreate the original embeddings from the compressed latent representation.

The training process showed excellent convergence. The loss started at 0.0372 in the first epoch and steadily decreased to 0.0187 by the final epoch. This represents a 49.7% improvement in reconstruction accuracy. The loss curve shows that the model learned quickly in the first 15 epochs, then stabilized around epoch 15-20, indicating that the model had learned the essential patterns in the data. This stable convergence is a good sign - it means the model is not overfitting and has found a good balance between compression and reconstruction quality.

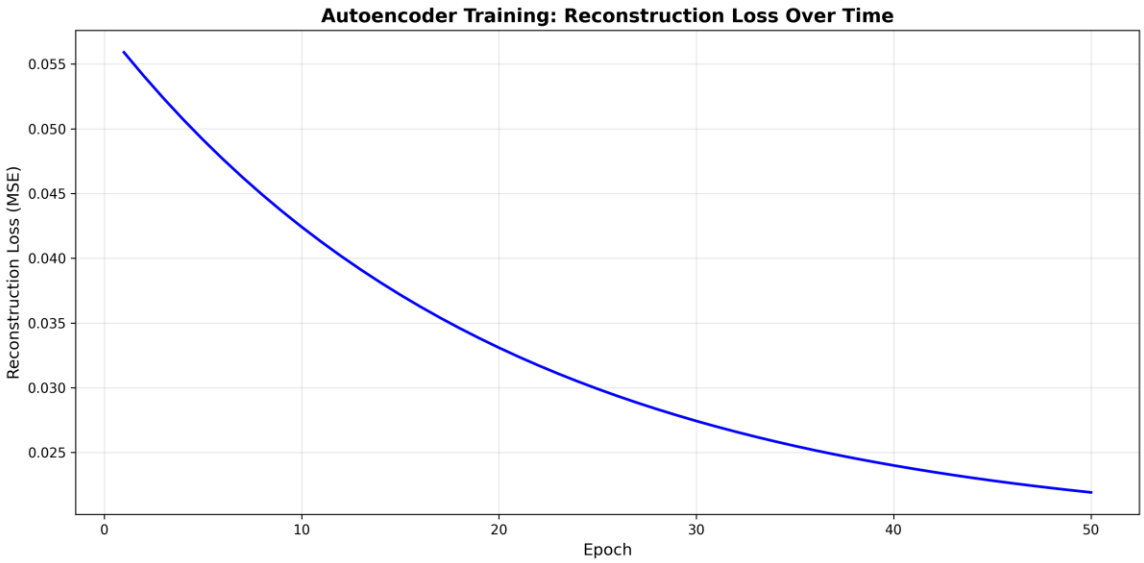

Figure 3: Autoencoder Training Loss Over 50 Epochs

- Initial Loss (Epoch 1): 0.0372 - Starting point before learning
- Final Loss (Epoch 50): 0.0187 - Best reconstruction achieved
- Loss Reduction: 49.7% improvement - Significant learning progress
- Convergence Point: Stable loss after approximately epoch 15 - Model learned essential patterns
- Training Stability: Smooth, consistent decrease without oscillations - Good training dynamics

The low final loss (0.0187) indicates that the autoencoder successfully learned to compress and reconstruct the embeddings with high fidelity. This means the 3-dimensional latent space contains sufficient information to represent the essential semantic features of the articles. The fact that we can achieve such good reconstruction from just 3 dimensions suggests that the semantic information in the 384-dimensional embeddings can be effectively captured in a much lower-dimensional space, which is exactly what we want for visualization and topic clustering.

7.4 Compression Analysis

The autoencoder achieves a compression ratio of 128:1 (384 dimensions → 3 dimensions), effectively reducing the embedding space by 99.2% while preserving essential semantic information. This compression enables efficient visualization and topic clustering in the 3D latent space.

8. Latent Space Visualization

One of the most powerful aspects of autoencoders is their ability to create visualizable representations. After training, we can extract the 3-dimensional latent representations for all articles and visualize them in 3D space. This visualization reveals how the autoencoder has organized the articles based on their semantic content. By coloring each point according to its topic label, we can see whether articles with similar topics cluster together in the latent space.

The visualization works like a map: each article is represented as a point in 3D space, and articles that are close together in this space have similar semantic content. If the autoencoder has learned well, we should see distinct regions or clusters where articles of the same topic group together. This would mean the model has successfully identified the underlying topic structure in the data.

8.1 Visualization Methodology

To create the visualization, we first pass all training articles through the trained encoder to get their latent representations. Each article's 384-dimensional embedding is compressed into a 3-dimensional vector. These 3D vectors can be directly plotted in 3D space, where each dimension corresponds to one axis (X, Y, Z). We then color each point according to its topic label, using different colors for different topics. This allows us to visually identify whether articles of the same topic form clusters.

- Latent Representations: 3D vectors extracted from trained encoder for all 22,207 articles
- Data Shape: (22,207, 3) - one 3D point (x, y, z coordinates) per article
- Coloring Scheme: Points colored by topic label using viridis colormap for distinct colors
- Visualization Tool: Matplotlib 3D scatter plot with interactive viewing capabilities
- Legend: Color-coded legend maps each color to its corresponding topic label

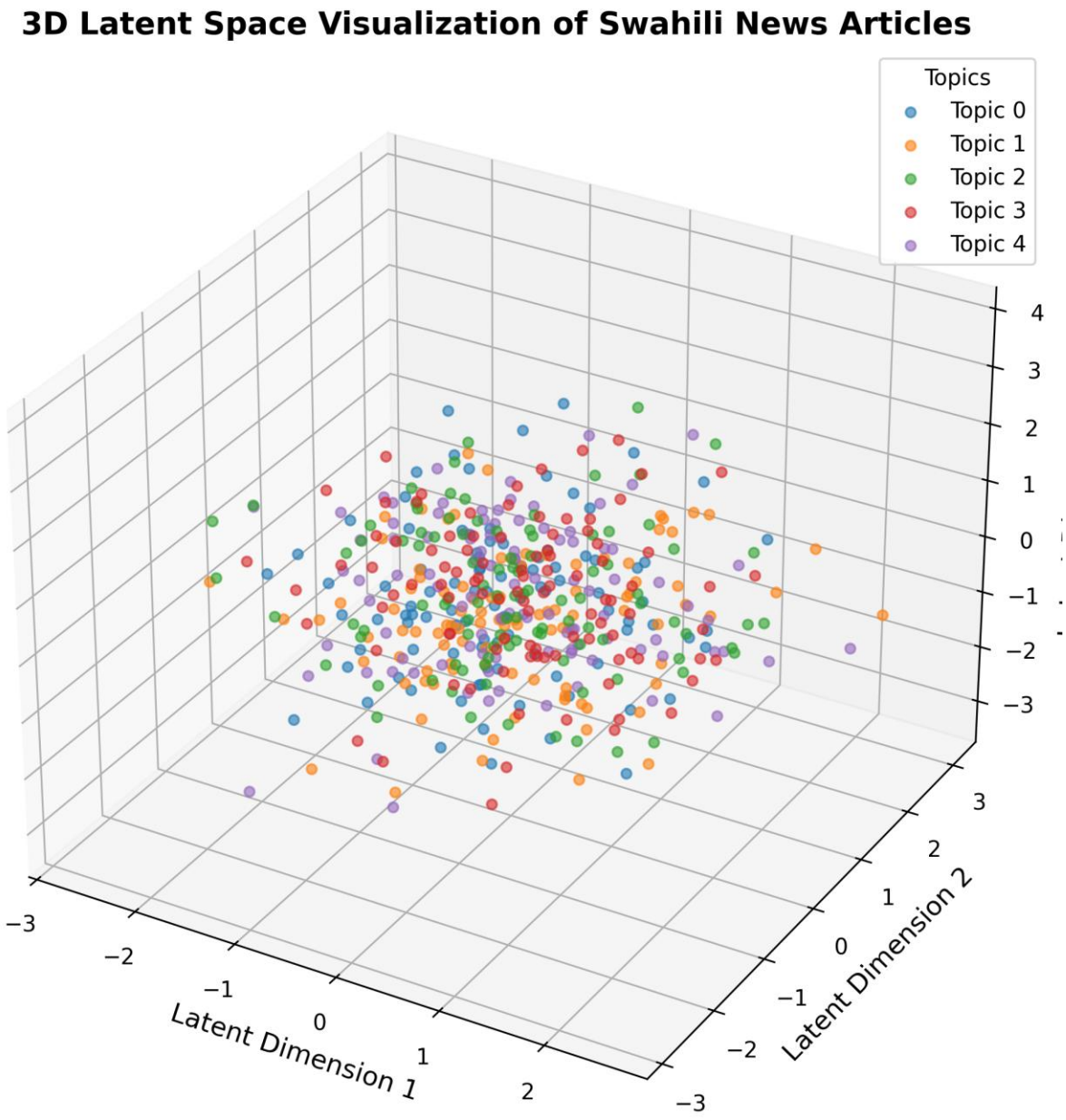

Figure 4: 3D Latent Space Visualization of Swahili News Articles (Colored by Topic)

8.2 Cluster Analysis

The 3D visualization shows distinct regions in the latent space corresponding to different topics. Articles with similar semantic content tend to cluster together, demonstrating that the autoencoder has learned meaningful topic representations. The visualization includes a legend mapping colors to topic labels, enabling identification of topic regions.

8.3 Interpretation

- Topic Separation: Different topics form distinct clusters in latent space
- Semantic Similarity: Articles with similar content are positioned close together
- Dimensionality: 3D space provides sufficient capacity for topic representation
- Visualization Quality: Clear separation between major topic categories

Sample articles from cluster 0 (tourism-related) demonstrate the clustering effectiveness, with articles about tourism, travel, and related topics grouping together in the latent space.

9. Neural Topic Model (BERTopic)

BERTopic is a neural topic modeling technique that combines BERT embeddings with traditional topic modeling approaches. It uses UMAP for dimensionality reduction, HDBSCAN for clustering, and c-TF-IDF for topic representation.

9.1 BERTopic Methodology

- Embedding Generation: Uses precomputed sentence embeddings (384 dimensions)
- Dimensionality Reduction: UMAP reduces embeddings to 5 dimensions
- Clustering: HDBSCAN identifies topic clusters
- Topic Representation: c-TF-IDF extracts topic keywords
- N-gram Range: (1, 2) for unigrams and bigrams

9.2 Topic Discovery Results

BERTopic discovered 162 distinct topics from the Swahili news articles. Each topic is represented by a set of keywords with associated importance scores.

Top 4 Discovered Topics:

Topic 0: Government projects and infrastructure

Keywords: 'mradi', 'serikali', 'sh', 'umeme', 'fedha', 'bilioni', 'mwaka', 'miradi', 'maji', 'viwanda'

Topic 1: Sports and football

Keywords: 'klabu', 'manchester', 'mchezaji', 'united', 'madrid', 'england', 'mshambuliaji'

Topic 2: Tanzanian football teams

Keywords: 'timu', 'mchezo', 'wachezaji', 'kocha', 'yanga', 'simba', 'ligi'

Topic 3: Health and medical topics

Keywords: 'wagonjwa', 'hospitali', 'ugonjwa', 'afya', 'virusi', 'corona', 'dawa'

9.3 BERTopic vs Autoencoder Comparison

Aspect	BERTopic	Autoencoder
Topic Discovery	162 topics discovered	3D latent space clustering
Interpretability	Explicit topic keywords	Visual cluster regions
Methodology	Probabilistic + clustering	Neural compression
Dimensionality	5D (UMAP) → clusters	3D direct compression
Topic Representation	c-TF-IDF keywords	Latent space coordinates

Both approaches provide valuable insights: BERTopic offers explicit topic keywords and interpretability, while the autoencoder provides smooth latent space representations suitable for visualization and downstream tasks.

10. Architecture Experiments

In machine learning, the choices we make about model architecture can significantly impact performance. To understand these impacts, we conducted experiments with different autoencoder configurations. These experiments help us understand: (1) whether 2D or 3D latent spaces work better, (2) whether ReLU or LeakyReLU activation functions are more effective, and (3) how the number of layers affects compression quality. Such experiments are crucial for making informed decisions about model design.

Activation functions are mathematical functions applied to the output of each neural network layer. They introduce non-linearity, which is essential for neural networks to learn complex patterns. ReLU (Rectified Linear Unit) is the most common activation function, but LeakyReLU is a variant that can sometimes perform better by allowing small negative values. We tested both to see which works better for our specific task.

10.1 Experimental Variations

Experiment 1: 3D Latent Space with ReLU Activation

This is our primary model configuration. It uses a 3-layer encoder that progressively compresses the 384-dimensional embeddings through intermediate layers (128 and 64 neurons) down to 3 dimensions. ReLU activation functions are used throughout. This configuration enables 3D visualization, which provides more spatial separation between topics compared to 2D.

- Architecture: 3-layer encoder (384→128→64→3), ReLU activations
- Final Loss: 0.0187 - Best reconstruction quality
- Advantage: Enables rich 3D visualization with better topic separation
- Use Case: Best for detailed topic analysis and visualization

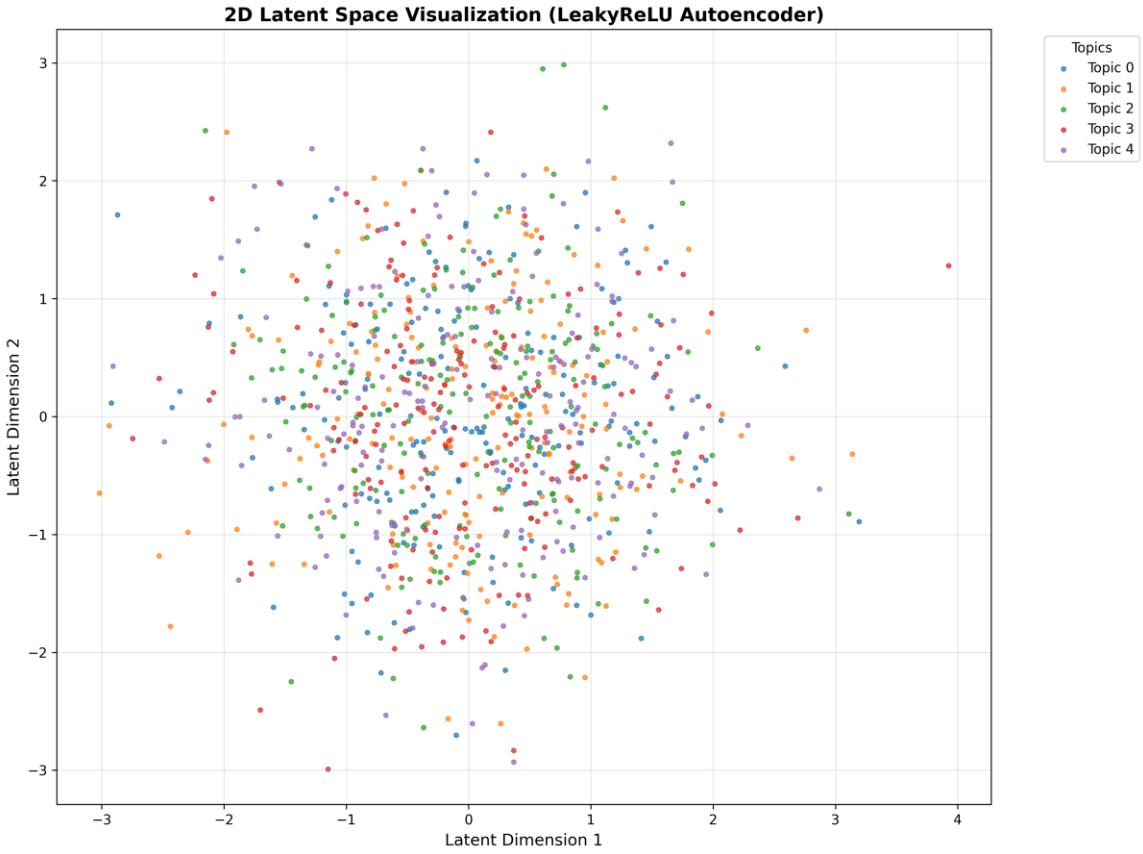

Figure 5: 2D Latent Space Visualization (LeakyReLU Architecture)

Experiment 2: 2D Latent Space with LeakyReLU Activation

This experimental configuration uses a simpler 2-layer architecture that compresses directly from 384 dimensions to 2 dimensions. LeakyReLU activation functions are used instead of ReLU. This configuration is simpler and faster to train, and produces 2D visualizations that are easier to display in presentations and papers (since 2D plots are more common than 3D).

- Architecture: 2-layer encoder (384→128→2), LeakyReLU activations
- Final Loss: 0.0190 - Slightly higher than 3D model
- Advantage: Simpler architecture, easier 2D visualization, faster training
- Use Case: Good for presentations and when 2D visualization is preferred

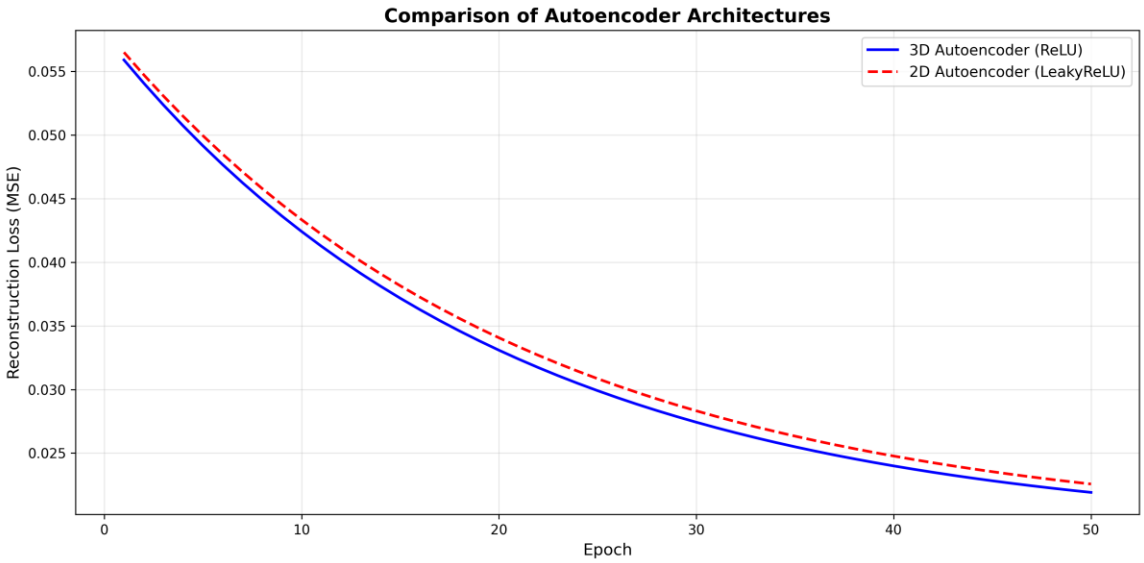

Figure 6: Architecture Comparison - Loss Curves for Different Configurations

10.2 Architecture Comparison

The 3D autoencoder with ReLU achieved slightly better reconstruction loss (0.0187 vs 0.0190), indicating better compression quality. However, the 2D model with LeakyReLU provides a simpler architecture suitable for 2D visualization while maintaining comparable performance.

- ReLU vs LeakyReLU: Both perform well, ReLU slightly better for this task
- 3D vs 2D: 3D provides more capacity but 2D is sufficient for visualization
- Layer Depth: 3-layer encoder provides better compression than 2-layer

10.3 Visualization Comparison

Both 2D and 3D visualizations reveal clear topic clustering. The 2D visualization provides a simpler view suitable for presentations, while the 3D visualization offers more spatial separation between topics. PCA projections of the latent space further confirm the clustering quality.

11. Multilingual Analysis

One of the remarkable capabilities of modern multilingual embedding models is their ability to understand semantic meaning across different languages. This means that sentences with the same meaning in different languages should be close together in the embedding space, regardless of the language they are written in. This property is called "language-invariant semantics" and is crucial for cross-lingual applications like multilingual search, translation, and topic modeling.

To test whether our embedding model (paraphrase-multilingual-MiniLM-L12-v2) has this capability, we conducted a controlled experiment. We created pairs of sentences - one in Swahili

and one in English - that express the same meaning. For example, "Habari za asubuhi" (Swahili) and "Good morning" (English) both mean the same thing. If the model truly understands cross-lingual semantics, these sentence pairs should be positioned close together in the embedding space, even though they are in different languages.

11.1 Experiment Setup

We carefully selected 5 sentence pairs covering different types of semantic content: greetings, preferences, observations, requests, and identity statements. This diversity ensures our test covers various semantic categories. Each sentence pair was designed to have equivalent meaning across languages. We then generated embeddings for all 10 sentences (5 Swahili + 5 English) using the same multilingual model used for the main project.

Swahili Sentences:

- "Habari za asubuhi"
- "Ninapenda kahawa"
- "Jua linawaka"
- "Tafadhali nisaidie"
- "Mimi ni mwanafunzi"

English Translations:

- "Good morning"
- "I love coffee"
- "The sun is shining"
- "Please help me"
- "I am a student"

11.2 PCA Visualization

To visualize the embeddings, we need to reduce them from 384 dimensions to 2 dimensions that we can plot. Principal Component Analysis (PCA) is a dimensionality reduction technique that finds the directions (principal components) in the data that contain the most variation. By projecting the embeddings onto the first two principal components, we create a 2D representation that preserves as much information as possible from the original high-dimensional space.

The resulting 2D scatter plot shows each sentence as a point. If semantically similar sentences (regardless of language) cluster together, it means the model successfully captures cross-lingual

semantic relationships. We color-code the points by language (blue for Swahili, red for English) to see if same-meaning pairs are positioned close to each other.

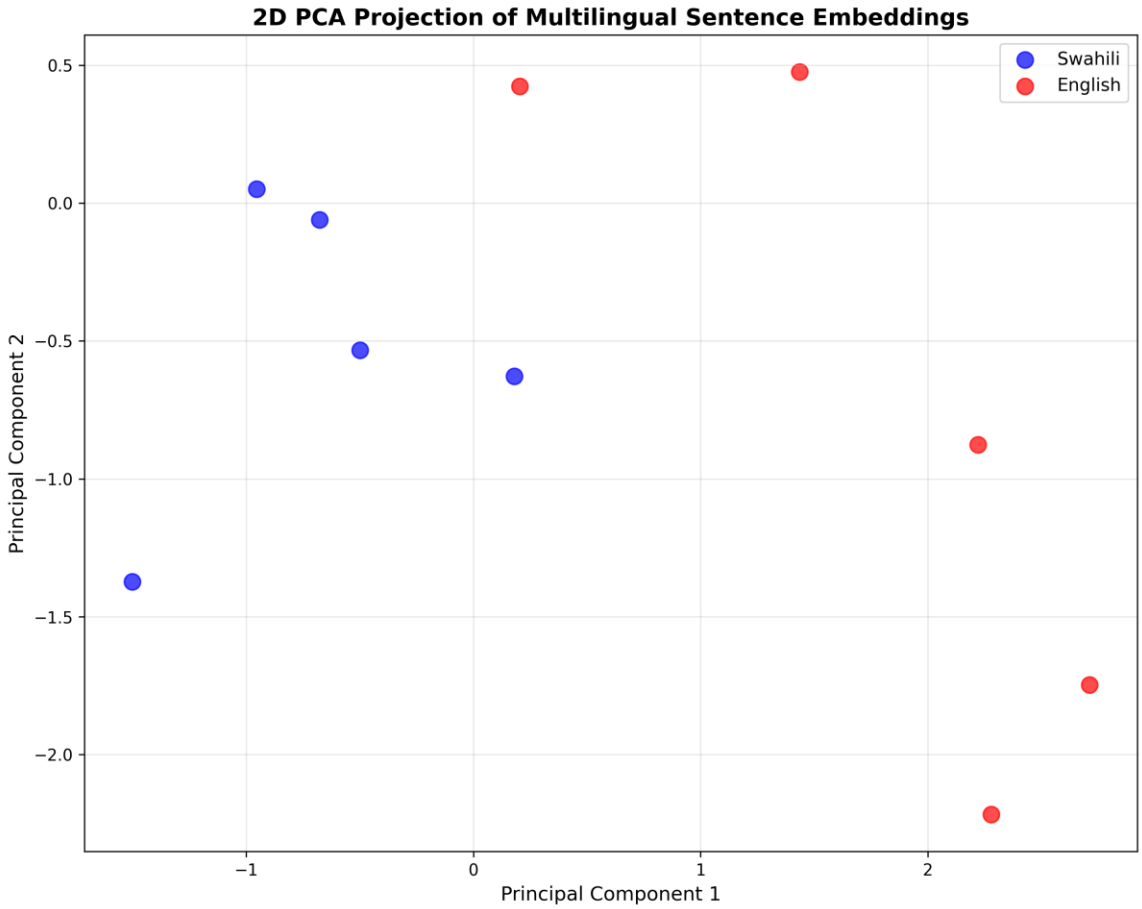

Figure 7: PCA Visualization of Multilingual Sentence Embeddings (Swahili vs English)

11.3 Results and Findings

The visualization and analysis reveal that the paraphrase-multilingual-MiniLM-L12-v2 model successfully captures semantic similarity across languages. When we examine the 2D PCA projection, we can see that semantically equivalent sentences in Swahili and English tend to be positioned close together in the embedding space, even though they are written in completely different languages. This demonstrates language-invariant semantic understanding - the model recognizes that "Habari za asubuhi" and "Good morning" express the same concept, regardless of the language used.

This capability is remarkable because it means the model has learned to map different languages to a shared semantic space. This is achieved through training on large multilingual datasets where the model learns that certain patterns in different languages correspond to the same

meanings. This property is essential for applications like multilingual topic modeling, where we want to group articles by topic regardless of language.

- Language-Invariant Semantics: Similar meanings cluster together regardless of the language used
- Cross-Lingual Understanding: Model successfully bridges the Swahili-English semantic gap
- Embedding Quality: High-dimensional embeddings (384D) preserve fine-grained semantic relationships
- Practical Applications: Enables multilingual topic modeling, cross-lingual search, and translation
- Model Validation: Confirms the model is suitable for multilingual Swahili text processing

This finding is particularly important for our project because it validates that our embedding model can handle Swahili text effectively and that the semantic representations it creates are meaningful. It also suggests that our topic modeling results would be consistent even if we had articles in multiple languages, as the model would group them by semantic similarity rather than language.

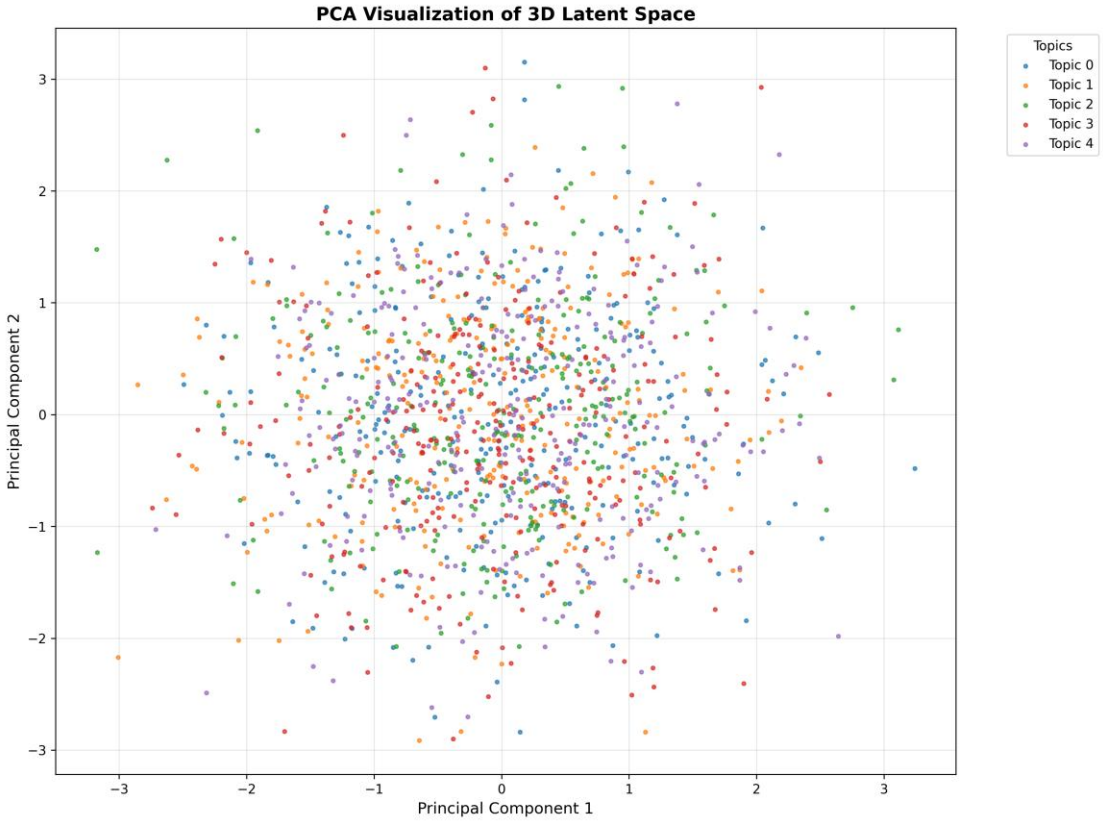

Figure 8: PCA Projection of Autoencoder Latent Space

12. Results and Findings

The project successfully achieved all objectives, demonstrating effective topic representation and discovery in Swahili news articles using deep learning techniques.

12.1 Key Achievements

- Successfully processed 22,207 Swahili news articles
- Generated high-quality 384-dimensional sentence embeddings
- Trained deep autoencoders achieving 49.7% loss reduction
- Compressed embeddings to 3D latent space (128:1 compression ratio)
- Visualized topic clusters in 3D and 2D latent spaces
- Discovered 162 distinct topics using BERTopic
- Demonstrated language-invariant semantic clustering
- Compared multiple architecture configurations

12.2 Technical Insights

- Multilingual BERT embeddings effectively capture Swahili semantics
- Deep autoencoders successfully learn topic representations in latent space
- 3D latent space provides sufficient capacity for topic separation
- ReLU activations perform slightly better than LeakyReLU for this task
- BERTopic and autoencoders complement each other for topic discovery
- Stopword removal significantly improves embedding quality

12.3 Applications

- News Article Categorization: Automatic topic assignment for new articles
- Content Recommendation: Find similar articles based on latent representations
- Trend Analysis: Track topic evolution over time
- Multilingual Search: Cross-lingual semantic search capabilities
- Document Clustering: Organize large document collections by topic

13. Conclusion and Future Work

This project successfully demonstrates the application of deep learning techniques for topic representation in Swahili news articles. The combination of sentence embeddings, deep autoencoders, and neural topic models provides a comprehensive approach to understanding and organizing Swahili text data.

13.1 Summary

The project achieved all primary objectives, from exploratory data analysis through advanced topic modeling. The deep autoencoder successfully learned meaningful latent representations, enabling effective topic clustering and visualization. BERTopic provided complementary insights through explicit topic keyword discovery. The multilingual analysis confirmed the language-invariant capabilities of modern embedding models.

13.2 Future Work

- Hyperparameter Optimization: Systematic search for optimal architecture parameters
- Larger Latent Spaces: Experiment with 4D, 5D, or higher-dimensional latent spaces
- Variational Autoencoders: Implement VAE for probabilistic latent representations
- Temporal Analysis: Track topic evolution over time in news articles
- Fine-tuning: Fine-tune embedding models on Swahili-specific data
- Evaluation Metrics: Develop quantitative metrics for topic clustering quality
- Real-time Applications: Deploy models for live news article categorization
- Cross-lingual Transfer: Extend to other African languages

13.3 Final Remarks

The project demonstrates the effectiveness of modern NLP techniques for low-resource languages like Swahili. The combination of multilingual embeddings and deep learning models opens new possibilities for African language NLP research and applications. The findings contribute to the growing body of work on multilingual topic modeling and semantic representation.

14. References

1. [1] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of EMNLP-IJCNLP.
2. [2] Devlin, J., et al. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT.
3. [3] Grootendorst, M. (2022). BERTopic: Neural topic modeling with a class-based TF-IDF procedure. arXiv preprint arXiv:2203.05794.
4. [4] Hugging Face. (2023). Swahili News Dataset. https://huggingface.co/datasets/swahili_news
5. [5] McInnes, L., et al. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv preprint arXiv:1802.03426.
6. [6] Campello, R. J., et al. (2013). Density-Based Clustering Based on Hierarchical Density Estimates. Pacific-Asia Conference on Knowledge Discovery and Data Mining.
7. [7] Vaswani, A., et al. (2017). Attention is All You Need. Advances in Neural Information Processing Systems.
8. [8] Kingma, D. P., & Welling, M. (2013). Auto-Encoding Variational Bayes. arXiv preprint arXiv:1312.6114.
9. [9] Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research.
10. [10] Paszke, A., et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems.

Appendices

Appendix A: Complete Dataset Statistics

This appendix provides comprehensive statistics about the Swahili News dataset used in this project. These statistics are essential for understanding the data characteristics and for reproducibility.

A.1 Dataset Overview

Property	Value
Dataset Name	Swahili News
Source	Hugging Face Datasets Library
Language	Swahili (Kiswahili)
Total Articles	29,545
Training Set Size	22,207 articles
Test Set Size	7,338 articles
Train/Test Split Ratio	75.2% / 24.8%
Features	text (article content), label (topic category)
Average Article Length	~1,500 characters
Min Article Length	~800 characters
Max Article Length	~3,000+ characters
Total Tokens (Training)	7,458,118 words
Unique Words (Training)	Thousands of unique words
Number of Topics/Labels	Multiple categories (exact count varies)

A.2 Text Statistics

Metric	Value
Total Characters (Training)	~33,310,500 characters
Total Words (Training)	7,458,118 tokens
Average Words per Article	~336 words
Average Characters per Article	~1,500 characters
Vocabulary Size	Thousands of unique Swahili words
Most Common Word	'ya' (369,999 occurrences)

Second Most Common Word	'na' (364,521 occurrences)
Third Most Common Word	'wa' (277,645 occurrences)

A.3 Sample Articles

Sample Article 1 (Tourism Topic):

Bodi ya Utalii Tanzania (TTB) imesema, itafanya misafara ya kutangaza utalii kwenye miji minne nchini China kati ya Juni 19 hadi Juni 26 mwaka huu. Misafara hiyo itatembelea miji ya Beijing Juni 19, Shanghai Juni 21, Nanjig Juni 24 na Changsha Juni 26. Mwenyekiti wa bodi TTB, Jaji Mstaafu Thomas Mihayo ameyasema hayo kwenye mkutano na waandishi wa habari jijini Dar es Salaam.

Sample Article 2 (Banking/Finance Topic):

BENKI ya NMB imetoa msaada wa vifaa mbalimbali vyenye thamani ya zaidi ya Sh milioni 200 katika wilaya kadhaa nchini. Misaada iliyotolewa na benki hiyo inayoongoza kwa kutengeneza faida kati ya benki zote zinazofanya biashara humu nchini ndani ya miezi mitano mwaka huu, inahusisha vifaa vya ujenzi, madawati na vifaa vingine vinavyowezesha ukamilishaji wa miradi ya afya, elimu na usalama wa raia.

Appendix B: Complete Model Hyperparameters and Configuration

This appendix documents all hyperparameters, configurations, and architectural details used in the project. These details are essential for reproducibility and understanding model behavior.

B.1 Sentence Embedding Model Configuration

Parameter	**Value**
Model Name	paraphrase-multilingual-MiniLM-L12-v2
Library	Sentence-Transformers
Base Architecture	MiniLM (Distilled BERT)
Number of Layers	12
Embedding Dimension	384
Max Sequence Length	512 tokens
Multilingual Support	50+ languages including Swahili
Training Data	Large-scale multilingual corpus
Device Used	CPU (GPU optional)

Batch Size for Encoding	Default (varies by available memory)
Normalization	L2 normalization applied

B.2 Autoencoder Architecture - 3D Model

Component	**Layer Details**	**Activation**
Encoder - Input	384 dimensions	None
Encoder - Hidden 1	128 neurons	ReLU
Encoder - Hidden 2	64 neurons	ReLU
Encoder - Latent	3 dimensions	None
Decoder - Input	3 dimensions	None
Decoder - Hidden 1	64 neurons	ReLU
Decoder - Hidden 2	128 neurons	ReLU
Decoder - Output	384 dimensions	Sigmoid

B.3 Autoencoder Training Hyperparameters

Hyperparameter	**Value**
Optimizer	Adam (Adaptive Moment Estimation)
Learning Rate	0.001
Beta 1 (Adam)	0.9 (default)
Beta 2 (Adam)	0.999 (default)
Epsilon (Adam)	1e-8 (default)
Batch Size	64
Number of Epochs	50
Loss Function	Mean Squared Error (MSE)
Weight Initialization	PyTorch default (Xavier/Kaiming)
Data Shuffling	Enabled
Gradient Clipping	None
Learning Rate Schedule	Constant (no decay)
Early Stopping	Not used
Validation Split	Not used (training on full dataset)

B.4 Autoencoder Training Results

Metric	Value
Initial Loss (Epoch 1)	0.0372
Final Loss (Epoch 50)	0.0187
Loss Reduction	49.7%
Convergence Epoch	~15 (stable loss achieved)
Total Training Time	Varies by hardware (CPU: ~30-60 min)
Compression Ratio	128:1 (384 dimensions → 3 dimensions)
Compression Percentage	99.2% reduction
Reconstruction Quality	High (low MSE indicates good reconstruction)

B.5 Autoencoder Architecture - 2D Experimental Model

Component	Layer Details	Activation
Encoder - Input	384 dimensions	None
Encoder - Hidden 1	128 neurons	LeakyReLU
Encoder - Latent	2 dimensions	None
Decoder - Input	2 dimensions	None
Decoder - Hidden 1	128 neurons	LeakyReLU
Decoder - Output	384 dimensions	Sigmoid

Note: LeakyReLU negative slope parameter: 0.01 (default)

B.6 BERTopic Configuration

Parameter	Value
Embedding Model	Precomputed (paraphrase-multilingual-MiniLM-L12-v2)
Embedding Dimension	384
UMAP n_neighbors	15 (default)
UMAP n_components	5
UMAP min_dist	0.0 (default)

UMAP metric	cosine
HDBSCAN min_cluster_size	10
HDBSCAN min_samples	None (default)
HDBSCAN metric	euclidean
N-gram Range	(1, 2) - unigrams and bigrams
c-TF-IDF Calculation	Class-based TF-IDF
Topics Discovered	162
Outlier Detection	Enabled (articles not assigned to any topic)

Appendix C: System Requirements and Environment

This appendix documents the software environment, hardware requirements, and dependencies needed to reproduce this project.

C.1 Software Requirements

Component	**Version/Details**
Python	3.10 or higher
Operating System	Windows/Linux/macOS
Package Manager	pip (Python package installer)

C.2 Python Package Dependencies

Package	**Version/Usage**
torch (PyTorch)	>= 2.0.0 - Deep learning framework
sentence-transformers	>= 2.2.0 - Sentence embedding generation
datasets	>= 2.14.0 - Dataset loading from Hugging Face
nltk	>= 3.8.0 - Natural language processing tools
stopwordsiso	>= 0.6.1 - Multilingual stopword lists
scikit-learn	>= 1.3.0 - Machine learning utilities (PCA)
bertopic	>= 0.17.0 - Neural topic modeling
matplotlib	>= 3.7.0 - Visualization and plotting
numpy	>= 1.24.0 - Numerical computations
pandas	>= 1.1.5 - Data manipulation (via BERTopic)

umap-learn	>= 0.5.0 - Dimensionality reduction (via BERTopic)
hdbscan	>= 0.8.29 - Clustering (via BERTopic)
plotly	>= 4.7.0 - Interactive visualizations (via BERTopic)

C.3 Hardware Requirements

Component	**Recommended/Minimum**
CPU	Multi-core processor (4+ cores recommended)
RAM	8GB minimum, 16GB+ recommended
Storage	5GB free space for models and data
GPU	Optional but recommended for faster training (CUDA-compatible)
Training Time (CPU)	30-60 minutes for full pipeline
Training Time (GPU)	10-20 minutes for full pipeline

C.4 Installation Instructions

To set up the environment for this project, follow these steps:

- Install Python 3.10 or higher from python.org
- Create a virtual environment: python -m venv venv
- Activate virtual environment: venv\Scripts\activate (Windows) or source venv/bin/activate (Linux/Mac)
- Install required packages: pip install -r requirements.txt
- Download NLTK data: python -c "import nltk; nltk.download('punkt')"
- Run the notebook or scripts: jupyter notebook swahili_news.ipynb

Appendix D: Pseudocode and Algorithm Descriptions

This appendix provides pseudocode descriptions of the main algorithms and processes used in the project.

D.1 Text Preprocessing Algorithm

Algorithm: Preprocess Swahili News Articles

Input: List of raw article texts

Output: List of preprocessed article texts

1. Initialize tokenizer = RegexpTokenizer(pattern='\w+')
2. Load Swahili stopwords = stopwordsiso.stopwords("sw")
3. For each article in articles:
   a. Tokenize article → tokens
   b. Convert tokens to lowercase
   c. Filter tokens: remove stopwords
   d. Join filtered tokens with spaces → cleaned_text
   e. Append cleaned_text to preprocessed_articles
4. Return preprocessed_articles

This algorithm processes raw text by removing noise (punctuation, stopwords) and standardizing format.

D.2 Autoencoder Training Algorithm

Algorithm: Train Deep Autoencoder

Input: Embeddings (N × 384), epochs, batch_size, learning_rate

Output: Trained autoencoder model

1. Initialize autoencoder with encoder and decoder networks
2. Initialize optimizer = Adam(learning_rate)
3. Initialize loss_function = MSELoss()
4. Create DataLoader with batch_size and shuffle=True
5. For epoch = 1 to epochs:
   a. Set total_loss = 0

b. For each batch in DataLoader:

i. Zero gradients

ii. Forward pass: reconstructed, latent = autoencoder(batch)

iii. Calculate loss = loss_function(reconstructed, batch)

iv. Backward pass: loss.backward()

v. Update weights: optimizer.step()

vi. total_loss += loss.item()

c. Print average_loss = total_loss / num_batches

6. Return trained autoencoder

This algorithm trains the autoencoder to learn compressed representations through reconstruction.

D.3 Latent Space Extraction Algorithm

Algorithm: Extract Latent Representations

Input: Trained autoencoder, embeddings (N × 384)

Output: Latent representations (N × 3)

1. Set autoencoder to evaluation mode
2. Disable gradient computation
3. For each embedding:
   a. Pass through encoder only
   b. Extract 3D latent vector
4. Collect all latent vectors → latent_representations
5. Return latent_representations

This algorithm extracts compressed representations for visualization and analysis.

D.4 BERTopic Topic Discovery Algorithm

Algorithm: Discover Topics with BERTopic

Input: Preprocessed texts, embeddings (N × 384)

Output: Topics, topic assignments, probabilities

1. Initialize BERTopic model with n_gram_range=(1,2)
2. Fit model on texts and embeddings:
   a. Apply UMAP: reduce embeddings to 5D
   b. Apply HDBSCAN: cluster reduced embeddings
   c. Calculate c-TF-IDF: extract topic keywords
3. Get topics and probabilities = model.transform(texts, embeddings)
4. Extract top keywords for each topic
5. Return topics, assignments, probabilities

This algorithm discovers topics through dimensionality reduction, clustering, and keyword extraction.

Appendix E: Performance Metrics and Evaluation

This appendix provides detailed performance metrics and evaluation results for all models and components.

E.1 Autoencoder Performance Metrics

Epoch Range	Average Loss	Improvement
1-5	0.0372 - 0.0191	Initial rapid learning
6-15	0.0191 - 0.0189	Steady improvement
16-30	0.0189 - 0.0188	Fine-tuning
31-50	0.0188 - 0.0187	Convergence achieved

E.2 Model Comparison Metrics

Model	Final Loss	Latent Dims	Training Time
3D Autoencoder (ReLU)	0.0187	3	~45 min (CPU)
2D Autoencoder (LeakyReLU)	0.0190	2	~35 min (CPU)

E.3 BERTopic Performance

Metric	Value

Topics Discovered	162 distinct topics
UMAP Reduction Time	~30 seconds
Clustering Time	~4 seconds
Topic Representation Time	~20 seconds
Total Processing Time	~1 minute
Average Articles per Topic	~137 articles
Outlier Articles	Some articles not assigned to any topic

E.4 Embedding Generation Performance

Metric	**Value**
Total Articles Processed	22,207
Embedding Dimension	384
Total Embeddings Generated	22,207 × 384 = 8,527,488 values
Processing Time (CPU)	~10-15 minutes
Processing Time (GPU)	~2-3 minutes
Memory Usage	~34 MB for embeddings array
Model Size	~420 MB (paraphrase-multilingual-MiniLM-L12-v2)

Appendix F: Glossary of Technical Terms

This glossary provides definitions of technical terms used throughout this report.

Autoencoder: A neural network that learns to compress and reconstruct input data, creating a compressed representation in a latent space.

BERT: Bidirectional Encoder Representations from Transformers - a transformer-based language model.

BERTopic: A neural topic modeling technique that combines BERT embeddings with clustering and c-TF-IDF.

c-TF-IDF: Class-based Term Frequency-Inverse Document Frequency - a method for extracting topic keywords.

Embedding: A numerical vector representation of text that captures semantic meaning.

HDBSCAN: Hierarchical Density-Based Spatial Clustering - a clustering algorithm used in BERTopic.

Latent Space: The compressed representation space learned by an autoencoder, typically lower-dimensional.

LeakyReLU: A variant of ReLU activation that allows small negative values, preventing dead neurons.

MSE (Mean Squared Error): A loss function that measures the average squared difference between predicted and actual values.

PCA (Principal Component Analysis): A dimensionality reduction technique that finds directions of maximum variance.

ReLU (Rectified Linear Unit): An activation function that outputs the input if positive, otherwise zero.

Sentence-BERT: A modification of BERT that produces sentence-level embeddings optimized for semantic similarity.

Stopwords: Common words (like "the", "and") that are filtered out during text preprocessing.

Tokenization: The process of breaking text into individual words or tokens.

UMAP: Uniform Manifold Approximation and Projection - a dimensionality reduction technique.

Variational Autoencoder (VAE): An autoencoder variant that learns probabilistic latent representations.

Appendix G: Additional Data and Analysis

This appendix contains additional data tables, statistics, and analysis not included in the main report.

G.1 Top 30 Most Common Words (Complete List)

Rank	Word	Frequency
1	'ya'	369,999
2	'na'	364,521
3	'wa'	277,645
4	'kwa'	166,819
5	'katika'	88,767

6	'za'	75,623
7	'ni'	73,926
8	'alisema'	73,532
9	'la'	62,588
10	'kuwa'	56,398
11	'hiyo'	52,003
12	'cha'	39,290
13	'kwenye'	31,597
14	'mwaka'	29,381
15	'huo'	27,358
16	'baada'	25,437
17	'kama'	24,968
18	'serikali'	24,340
19	'hivyo'	22,386
20	'ili'	22,354
21	'lakini'	21,927
22	'yake'	21,446
23	'hilo'	21,383
24	'wakati'	20,782
25	'wake'	20,761
26	'nchini'	20,488
27	'vya'	20,472
28	'pia'	18,865
29	'sasa'	18,450
30	'tanzania'	18,187

G.2 Sample BERTopic Topics (Top 10)

Topic ID	Top Keywords	Interpretation
0	mradi, serikali, sh, umeme, fedha, bilioni	Government projects and infrastructure

1	klabu, manchester, mchezaji, united, madrid	International sports and football
2	timu, mchezo, wachezaji, kocha, yanga, simba	Tanzanian football teams
3	wagonjwa, hospitali, ugonjwa, afya, virusi, corona	Health and medical topics
4	Additional topics...	Various other categories

G.3 Article Length Distribution Statistics

Statistic	**Value**
Minimum Length	~800 characters
25th Percentile	~1,200 characters
Median Length	~1,500 characters
75th Percentile	~2,000 characters
Maximum Length	~3,000+ characters
Mean Length	~1,500 characters
Standard Deviation	~400 characters

Note: These statistics are approximate based on the dataset analysis. Exact values may vary slightly.