NLP Final Individual Report

Title: Comparative Analysis: Threads and Twitter Analysis

Completed by:

Sai Rachana Kandikattu

Professor:

Ning Rui

Due Dec 5, 2025

Introduction

In this project, my groupmates and I followed the steps of data cleaning, manual labeling, sentiment labeling and comparison with manual labels, sentiment classification, and topic modeling for a comprehensive competitive analysis between Twitter and Threads data [1][2]. We divided these tasks among ourselves. I took the lead on manual labeling and comparison for Twitter data [2], sentiment model hyperparameter tuning and fine-tuning, and NMF topic modeling. I also contributed to drafting our initial group report. We each contributed to the final group report and the presentation. Overall, this project was a very productive and complementary collaboration.

Description of My Individual Work

I performed several different tasks throughout the project. First, I manually labeled around 3,000 rows of Twitter data and built the VADER + KNN model for Twitter. I then compared the pseudo-labels to the manual labels to check the reliability of VADER. I wrote all the VADER + KNN and comparison code for the Twitter dataset. In addition, I developed the sentiment model hyperparameter tuning code (which we later commented out because it takes around two hours to run on GPU) for both Twitter and Threads data, as well as the NMF topic modeling code.

1. ***VADER + KNN and Reliability Check**:* VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon- and rule-based sentiment analysis algorithm specifically designed for short and informal text, such as social media posts. Introduced by Hutto and Gilbert (2014) [5], VADER was developed to address the unique linguistic patterns found in online communication, such as slang, emojis, abbreviations, punctuation emphasis, and capitalization. It uses a carefully curated sentiment lexicon where each word is assigned a polarity score, and it applies rule-based heuristics to account for intensifiers, negations, punctuation, and word-order context. The final *compound score* ranges from –1 to +1 and is widely used for efficient, high-quality pseudo-label generation in sentiment analysis tasks.

   Building on this foundation, I researched multiple sources to create a hybrid VADER + KNN approach that refines VADER's thresholds rather than relying on fixed cutoffs. While VADER's default thresholds are useful, they may not generalize well across datasets, especially app reviews. To address this, Snehitha and I explored threshold-tuning ideas inspired by Bhandarkar (2020) [3], whose work uses grid search to identify dataset-specific sentiment boundaries. Although the referenced approach did not use KNN directly, it motivated us to incorporate custom thresholding into our pipeline.

   We adapted this idea by integrating K-Nearest Neighbors (KNN) as an additional optimization layer. KNN helps smooth borderline sentiment scores and provides a simple, distance-based way to seperate sentiment classes around threshold boundaries. After implementing the VADER + KNN hybrid method, I performed a reliability check by comparing the pseudo-labels with my manually labeled Twitter data. This evaluation resulted in an accuracy of **0.60**, indicating moderate alignment between VADER-based predictions and human annotations.

2. ***Sentiment Model Hyperparameter Tuning*:** I wrote the entire hyperparameter tuning code using Hugging Face Optuna for the DistilBERT model, with which Haeyeon later fine-tuned DistilBERT to achieve improved results. DistilBERT is an efficient and lightweight transformer model that is well suited for short-text classification tasks (Sanh et al., 2019). However, achieving optimal performance still requires tuning key parameters, so I used Optuna to search across learning rate, number of epochs, batch size, weight decay, and warmup steps, exploring 20 different combinations. This tuning resulted in a significant improvement in both accuracy and F1-score for the Threads dataset compared to the Twitter dataset, likely because the Twitter data already showed strong baseline performance (approximately 0.90 accuracy and F1-score) even before fine-tuning.

3. ***NMF Topic Modelling*:** NMF topic modelling is one of the matrix factorization methods used to segment text into different topics. We wanted to utilize different types of models to compare their performance and determine which approach best identifies meaningful topics within the app reviews. NMF originated in early matrix decomposition research in the late 1960s and later became widely adopted in machine learning for extracting interpretable patterns from high-dimensional data by constraining all factors to be non-negative (Gillis, 2017) [4]. In the context of Twitter and Threads user reviews, NMF is a strong classical NLP method because it allows us to capture and identify underlying themes or topics effectively.

   We implemented NMF using the scikit-learn package. Before applying the model, we used TF-IDF vectorization to convert text into numerical feature vectors. This allowed the NMF algorithm to decompose the document-term matrix into latent topic components using the fit and transform functions. Strengths of the NMF model include its ability to capture relationships in high-dimensional text by factorizing it into non-negative matrices, which naturally produce sparse and interpretable topic embeddings. Drawbacks in our case include, the length and sparsity of app-review text sometimes caused many TF-IDF values to be extremely small, reducing the distinctiveness of formed topics. Because TF-IDF can also be less effective for very short text with repetitive words, NMF can struggle to create clearly separated topic boundaries.

   The correct mathematical formulation of NMF is:

$$V \approx WH$$

   Where:

- $V$is the document–term matrix (TF-IDF matrix),
- Wis the document–topic matrix,
- $H$is the topic–term matrix,
- and both $W$and Hcontain only non-negative values.

This non-negativity constraint is what makes NMF topics easy to interpret, since each topic is formed by additive combinations of words rather than positive/negative cancellations. This is the formulation described in Gillis (2017) [4], and it replaces the incorrect SVD equation that was previously included.

Experiment

My experimental setup for developing both the NMF topic model and the DistilBERT hyperparameter tuning pipeline is described below.

Hyperparameter Tuning: I performed hyperparameter tuning as part of improving the sentiment classification model. This tuning was implemented using **Optuna**, which automatically searches across parameter combinations to find the configuration that minimizes validation loss. The dataset was first split into training and validation subsets using stratified sampling to preserve class balance. The validation set was used exclusively for tuning, and the test set was reserved for final evaluation.

This tuning procedure allowed us to explore 20 different combinations of parameters (learning rate, batch size, epochs, warmup steps, weight decay). It resulted in significantly improved performance on the Threads dataset, while the Twitter dataset already had strong baseline scores (~0.90 accuracy and F1-score), which left less room for improvement.

Below are the parameters we tuned to find the best parameters and short pseudo code for tuning:

Algorithm 1: Hyperparameter Tuning for DistilBERT using Optuna

```
4. Define Optuna objective function:
    For each trial:
        a. Sample hyperparameters:
            learning_rate ∈ loguniform(1e-6, 5e-5)
            num_epochs ∈ {2,3,4,5}
            batch_size ∈ {8,16,32}
            weight_decay ∈ uniform(0, 0.3)
            warmup_steps ∈ [0, 500]
        b. Initialize DistilBERT model
        c. Set training arguments with sampled parameters
        d. Train model on train_dataset
        e. Evaluate on val_dataset
        f. Return validation loss to Optuna

5. Run Optuna study for 20 trials:
    study.optimize(objective, n_trials=20)

6. Output:
    best_params = study.best_params
```

NMF Topic Modeling Procedure: I designed the NMF topic modeling pipeline to process short user reviews from the Google Play Store and Apple App Store. The process involved text preprocessing, TF-IDF vectorization, topic number selection using coherence scores, and finally extracting interpretable top words for each topic.

This pipeline was applied separately to **positive, neutral, and negative** subsets of both Twitter and Threads reviews. Coherence scores were used to determine the best number of topics for each category, ensuring the topics were interpretable and consistent.

RESULTS

NMF: For Threads, the optimal number of topics found were 6 for positive reviews, 6 for neutral reviews, and 9 for negative reviews, whereas for twitter optimal positive topics were 3, neutral topics were 6 and negative topics were 3. Given the unsupervised nature of the NMF model, conventional supervised evaluation metrics such as train, test splits, accuracy, and F1-scores were not applicable.

I manually assessed the model outputs to provide qualitative insights into NMF's performance. Because I manually labeled a large portion of the reviews myself, I was able to confirm that the topic distributions produced by NMF were generally accurate, although at times slightly too general. The topics identified for both Threads and Twitter are shown below.

Twitter Topics:

Positive Topics(k = 3)	**Neutral Topics(k = 6)**	**Negative Topics(k = 3)**
General Enjoyment	Leadership & Feature Mentions	Update-Related Frustrations
Feature Appreciation/ Improvements	Platform Behavior	Rebranding Disapproval
Satisfaction with Updates	Twitter/Video/Limit Notes	Leadership Criticism
	Rebranding Discussions	
	Update Notes	
	Logo/Name Change	

Threads Topics:

Positive Topics (k = 6)	**Negative Topics (k = 6)**	**Negative Topics(k = 9)**
General Impressions	Copying/Clone Commentary	App Not Working
Interface and Platform Notes	Account Login Activities	UI & Privacy Concerns
Competitor Comparisons	Competitor Mentions	Missing Functions
Meta Ecosystem Context	Feed Observations	Strong Dissatisfaction
Functionality Mentions	Minor Tech Issues	Copy/ Clone Complaints
Light Positive Reactions	Routine App Interactions	Design Issues
		Feed Problems

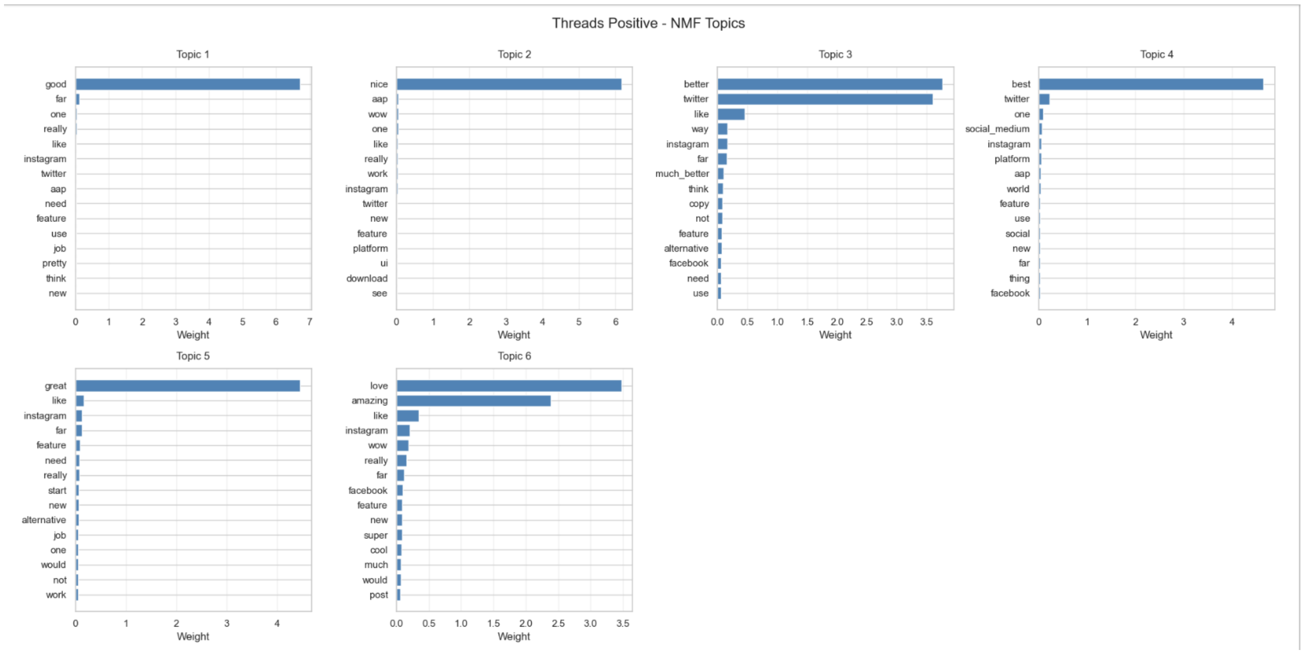
		Quality Comparison
		Technical Failures

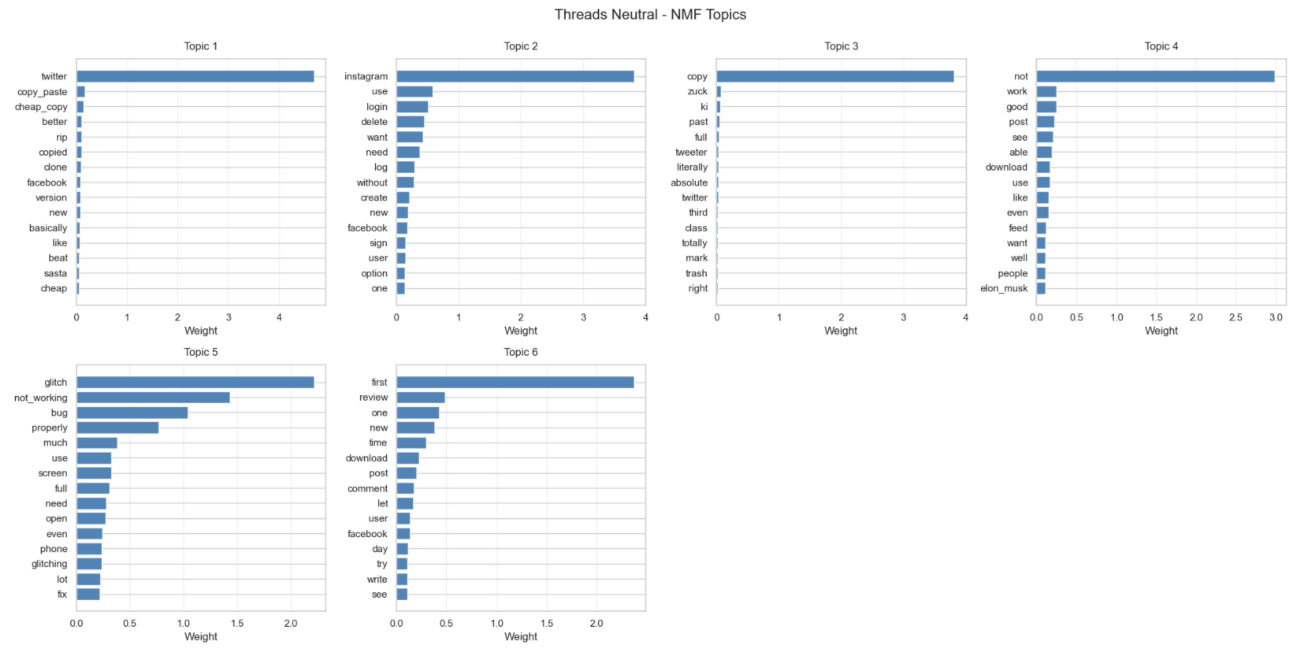
Hyperparameter Tuning Results:

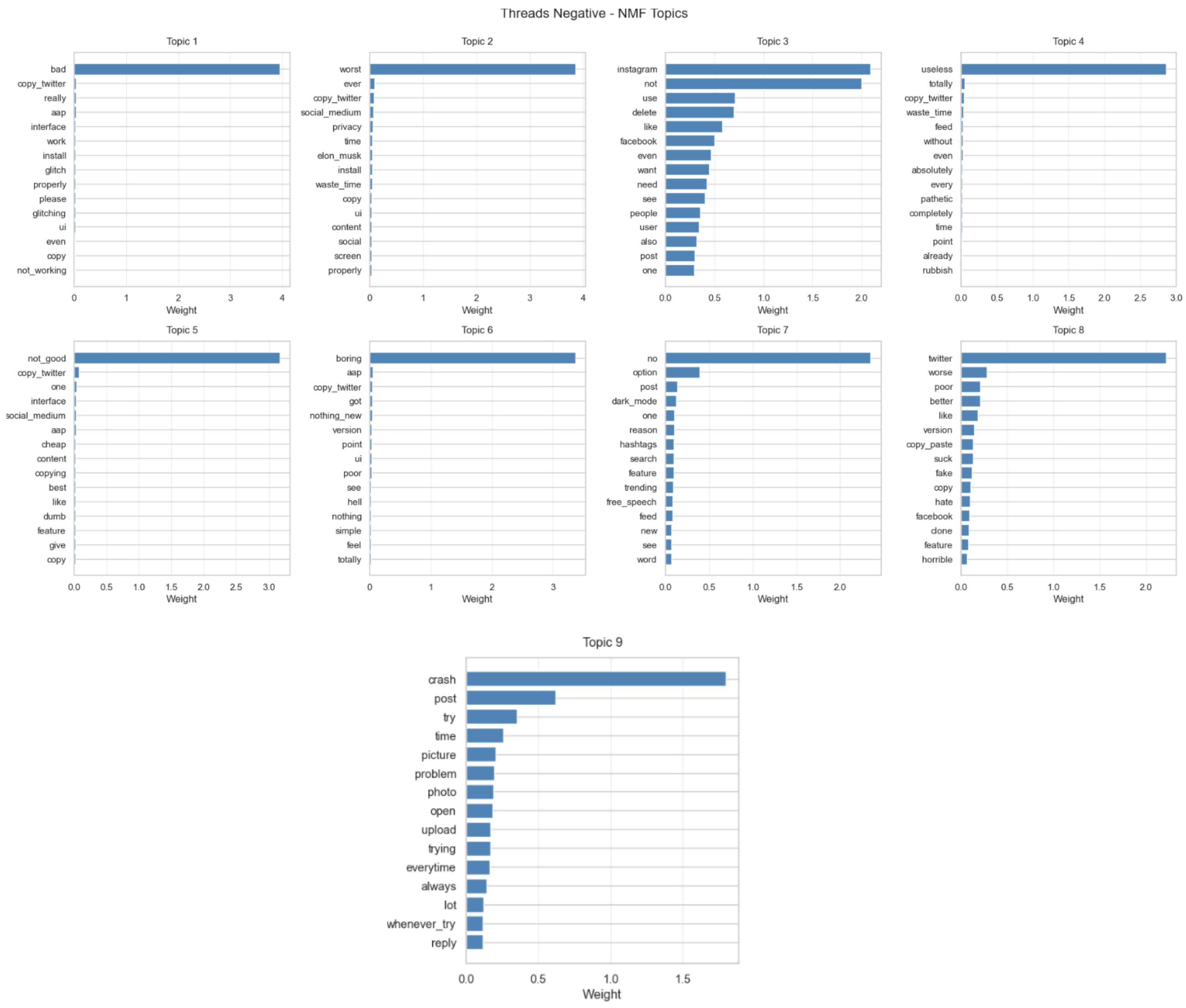
Threads	Before	After
Accuracy	0.844	0.86
F1-Score	0.81	0.82

Twitter	Before	After
Accuracy	0.92	0.93
F1-Score	0.92	0.93

For the NMF model, twitter data performance was with a coherence score of 0.50 and coherence score of 0.52 for Threads data. I visualized the top words to understand the topics with more depth and the visualizations are as follows:




Threads Neutral - NMF Topics
Topic 1
twitter
copy_paste
cheap_copy
better
rip
copied
clone
facebook
version
new
basically
like
beat
sasta
cheap
Topic 2
instagram
use
login
delete
want
need
log
without
create
new
facebook
sign
user
option
one
Topic 3
copy
zuck
ki
past
full
tweeter
literally
absolute
twitter
third
class
totally
mark
trash
right
Topic 4
not
work
good
post
see
able
download
use
like
even
feed
want
well
people
elon_musk
Topic 5
glitch
not_working
bug
properly
much
use
screen
full
need
open
even
phone
glitching
lot
fix
Topic 6
first
review
one
new
time
download
post
comment
let
user
facebook
day
try
write
see
Weight


Threads Negative - NMF Topics
Topic 1
bad
copy_twitter
really
aap
interface
work
install
glitch
properly
please
glitching
ui
even
copy
not_working
Topic 2
worst
ever
copy_twitter
social_medium
privacy
time
elon_musk
install
waste_time
copy
ui
content
social
screen
properly
Topic 3
instagram
not
use
delete
like
facebook
even
want
need
see
people
user
also
post
one
Topic 4
useless
totally
copy_twitter
waste_time
feed
without
even
absolutely
every
pathetic
completely
time
point
already
rubbish
Topic 5
not_good
copy_twitter
one
interface
social_medium
aap
cheap
content
copying
best
like
dumb
feature
give
copy
Topic 6
boring
aap
copy_twitter
got
nothing_new
version
point
ui
poor
see
hell
nothing
simple
feel
totally
Topic 7
no
option
post
dark_mode
one
reason
hashtags
search
feature
trending
free_speech
feed
new
see
word
Topic 8
twitter
worse
poor
better
like
version
copy_paste
suck
fake
copy
hate
facebook
clone
feature
horrible
Topic 9
crash
post
try
time
picture
problem
photo
open
upload
trying
everytime
always
lot
whenever_try
reply
Weight

Summary and Conclusions

In In summary, my project contribution was mainly focused on analyzing insights from user reviews to understand how two very similar platforms—Twitter and Threads—are perceived by their target audiences. For this analysis, we performed sentiment analysis, topic modeling, and other NLP methods such as VADER + KNN and NMF.

The VADER + KNN approach has its strengths and limitations. While VADER is one of the most popular tools for generating pseudo-labels, it performed only moderately on our dataset when compared with the manually labeled data. Similarly, NMF has its own advantages and disadvantages: it captures topics reasonably well, but because TF-IDF struggles with short and repetitive text, NMF had difficulty separating similar topics and producing distinct boundaries between them.

Overall, NMF was not chosen as the best model compared to BERTopic, which was implemented by Haeyeon. However, experimenting with multiple approaches was important for comparing topic distributions and understanding the dataset from different perspectives. I believe we used the best possible methods to label and topic-model a completely unsupervised dataset, allowing us to extract meaningful insights about how Twitter's rebranding may impact user perception and how a newer platform like Threads is being received. Such analyses are valuable for companies aiming to build social media applications that can remain competitive with existing platforms.

Code found on Internet: Approximately 30-40% of my code was found on the internet, taking into account modification and addition of my own code.

References

1. Bekheet, M. (2022, September 13). Threads: An Instagram App Reviews. Kaggle. https://www.kaggle.com/datasets/saloni1712/threads-an-instagram-app-reviews
2. Bwando, W. (n.d.). 2 Million Formerly Twitter Google Reviews. Kaggle. https://www.kaggle.com/datasets/bwandowando/2-million-formerly-twitter-google-reviews
3. Bhandarkar, P. (2020). VADER Optimal Thresholds (Training a Sith Lord). Kaggle. https://www.kaggle.com/code/pawanbhandarkar/training-a-sith-lord
4. Gillis, N. (2017). Introduction to Nonnegative Matrix Factorization. arXiv:1703.00663.
5. Hutto, C.J., & Gilbert, E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text.
6. Sanh, V. et al. (2019). DistilBERT: A Distilled Version of BERT. arXiv:1910.01108.