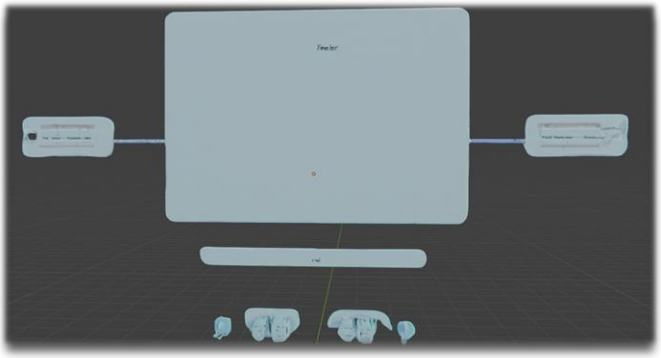
Feeler: An AI-Powered Sentiment Analysis Platform for Customer Feedback with NLP

Dr. Lawrence Nderu, Joram Kireki, Jany Muong, Vincent Ochieng', Gatmach Yuol, Akech Atem, and Josphat Waweru
Department of Computing, Jomo Kenyatta University of Agriculture and Technology

lnderu@jkuat.ac.ke, joramkireki@gmail.com

ABSTRACT:
Customer feedback analysis is critical for businesses, institutions and firms to improve products and services, but existing analysis techniques; most likely manual ones are inefficient and unscalable. This paper presents Feeler, an AI-powered sentiment analysis platform that leverages state-of-the-art Natural Language Processing (NLP) models to automatically classify customer feedback into sentiment categories. Feeler leverages NLP to analyze customer feedback in real time. It provides actionable insights, enabling us to understand customer sentiment, identify pain points, and make data-driven decisions. The platform supports natural language and offers granular sentiment classification ranging from e.g. very negative to very positive. The system integrates with target applications and with social media APIs, to processes text data efficiently, and provides actionable insights through an interactive dashboard. We package this system as a platform inside Django for use. We evaluate Feeler's performance using real-world datasets, demonstrating 93% accuracy in sentiment classification for natural language text. The platform's commercial viability is supported by a subscription-based SaaS model, targeting SMEs and enterprises. This work contributes to the field by optimizing accuracy, speed, and usability for English-language sentiment analysis.

Keywords: **Feeler, Sentiment Analysis, NLP, Customer Feedback, SaaS Platform, Text Processing, Text Vectorization, Tokenization**



1 INTRODUCTION

1.1 PROBLEM STATEMENT

Businesses receive **vast amounts of unstructured feedback** from social media, reviews, and surveys. Manual analysis (some of which is just someone judging text in their head, at least in the Kenyan context) is:

Time-consuming: hours spent categorizing feedback.

Inconsistent: human bias affects sentiment interpretation.

Limited in scalability: impossible for large datasets.

Also, existing tools (e.g., Brandwatch - a social media monitoring and analytics platform with sentiment analysis capabilities) are often prohibitively expensive for small businesses.

1.2 OUR CONTRIBUTION:

At the core of our approach are these questions, which have massive potential to guide our work:

1. How can AI-driven sentiment analysis models be optimized to accurately classify customer feedback across various industries while maintaining low latency?
2. What role do transformer-based NLP models play in improving sentiment classification accuracy compared to traditional rule-based or lexicon-based approaches?
3. How can Feeler integrate real-time sentiment analysis with social media and customer feedback platforms to provide actionable insights for

businesses?

4. What are the key challenges in designing an intuitive and scalable dashboard that effectively visualizes sentiment trends and user feedback?
5. How can sentiment analysis insights be leveraged to enhance customer engagement strategies and inform data-driven business decisions?
6. What trade-offs exist between model accuracy, inference speed, and computational cost in a commercial sentiment analysis application?
7. How can sentiment analysis models be fine-tuned for specific business domains to improve contextual understanding and classification reliability?

We have addresses part of these questions using Feeler (please see the literature review segment for this).

2 Literature Review

We introduce a platform to extract arbitrary sentiments from user prompts/social media posts within the parameters of Natural Language Processing techniques and pipelines. While this sentiment analysis platform is generalizable to any problem that could use language consolidation, we demonstrate its efficacy in solving customer feedback problems.

Current work on sentiment analysis and some of it worldwide often utilizes traditional feedback and sentiment detection models, which focus on broad population-level data and may not capture the intricacies of individual interactions within business product settings. While these models provide valuable insights, they fall short in dynamically detecting sentiment. Feeler offers a more fine-tuned approach by presenting everything in one place, while still capturing the complexity of customer emotion. Integrating AI and NLP with good user interfaces has the potential to enhance predictive accuracy and optimize sentiment analysis.

Here are some of the existing solutions/past work using approaches, including rule-based, machine learning, aspect-based sentiment analysis (ABSA) and hybrid methods, to analyze text and determine the underlying sentiment:

Tool	Accuracy	Speed	Limitations
VADER	72%	fast	rule-based, limited nuance
TextBlob	75%	fast	basic ML, low accuracy
BERT	89%	Slow	computational overhead

Table 1: existing solutions

Despite the advances in sentiment analysis techniques, there remains gaps in integrating NLP with systems that can effectively handle customer feedback. Feeler, and with AI on top of it, offers a powerful tool to bridge these gaps. Feeler addresses these gaps by:

1. Providing **high-accuracy sentiment analysis** optimized for English text.
2. Offering **granular sentiment classification** (positive/negative/ with confidence scores).
3. Delivering **real-time insights** via an intuitive dashboard.
4. Using **domain-specific fine-tuning** for business contexts.

Technical foundations: we are building our foundations on the working of these tools – we have seen how they work, and have, in one way or the other, inspired how **Feeler works**:

- **Distillation Techniques**: DistilBERT (Sanh et al., 2020) which provides faster inference with minimal accuracy loss.
- **Transformer Models**: BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) enable advanced NLP.

We expand on **Feeler** more in the following sections.

3 Methodology

3.1 NLP Background

Our methodology/approach is two-part - specialized models for robust sentiment analysis: a custom-trained Sentient73 model and the pre-trained Twitter-roBERTa-base transformer. This dual-model architecture ensures both high accuracy and computational efficiency.:

— **Sentient73**: a model trained on the Sentimen140 dataset ~1.6M tweets using **TensorFlow** and NLP techniques. Tweets were Preprocessed to handle Twitter-specific noise (hashtags, mentions, emojis etc).

— **Twitter-roBERTa-base**: for Sentiment Analysis: this is a roBERTa-base model trained on ~58M tweets and finetuned for sentiment analysis with the TweetEval benchmark. This model is suitable for English Natural Language Processing Tasks.
Output: 3-class sentiment (negative/neutral/positive)

Advantages: state-of-the-art performance on social media text, and it handles complex linguistic patterns (sarcasm, slang)

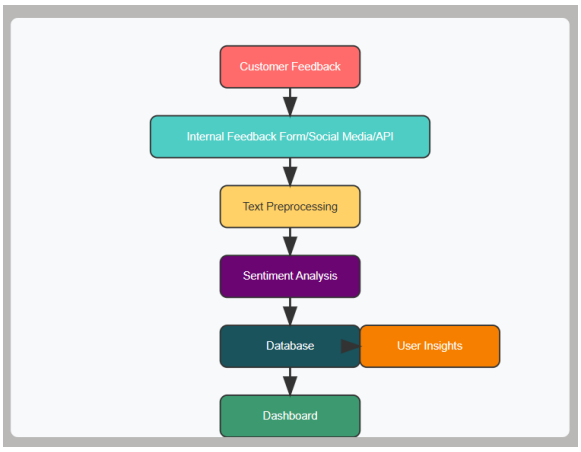
Reference Paper: TweetEval (Findings of EMNLP 2020).

3.2 JUSTIFICATION FOR METHODOLOGY

Criterion	Sentient73	Twitter-roBERTa	Combined Benefit
Speed	Faster (CPU-compatible)	Slower (requires GPU)	Use Sentient73 for real-time batch processing
Accuracy	Good (93.6%)	Excellent (94.1% on TweetEval)	Fallback to roBERTa for ambiguous cases
Resource Usage	Low (~50MB model size)	High (~500MB)	Cost-effective scaling
Output Granularity	Binary (pos/neg)	3-class (pos/neg/neutral)	Enables nuanced analysis

Table 2: justification for methodology

3.3 SYSTEM ARCHITECTURE



feeler figure 1: system arch

3.4 ALGORITHM FOR CUSTOM MODEL/SENTIENT73.KERAS

Sentient73 Training Pseudocode

1. LOAD DATASET
- Read CSV (training_cleaned.csv)
- Map columns: [polarity, id, date, query, user, text]

2. BUILD MODEL: Sequential Architecture
1. TextVectorization (10,000 tokens)
2. Embedding (128-dim)
3. Conv1D (128 filters, kernel=5)
4. GlobalMaxPooling1D()
5. Dense(64, ReLU)
6. Dropout(0.4)
7. Dense(1, Sigmoid)

- Compile: Adam optimizer, BinaryCrossentropy

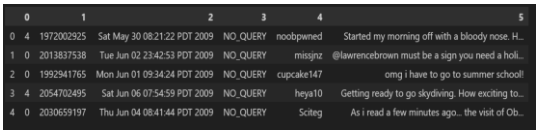
3. TRAIN
- Adapt vectorizer to text vocabulary
- Fit with:
  - 20 epochs
  - Batch size=64
  - Validation split=0.2

4. SAVE
- Export model as sentient73.keras
- Save vocabulary to vectorizer_vocab.txt

feeler figure 2: algorithm for sentient73

Select Algorithmic Choices/Architecture Techniques:

1. **Data Collection and Pre-processing**: the dataset input for this study is obtained from the **Sentiment140** dataset, we customized and cleaned it to fit our model architecture: e.g. empty entries (deleted tweets) were removed from the dataset, we removed all stopwords (i.e. common words such as "a", "an", "the", "and", etc.) and punctuation in the dataset. The final dataset looks like this:



	0	1	2	3	4	5
0	4	1972002925	Sat May 30 08:21:22 PDT 2009	NO_QUERY	noobpwned	Started my morning off with a bloody nose. H...
1	0	2013837538	Tue Jun 02 23:42:53 PDT 2009	NO_QUERY	missjnz	@lawrencebrown must be a sign you need a holi...
2	0	1992941765	Mon Jun 01 09:34:24 PDT 2009	NO_QUERY	cupcake147	omg i have to go to summer school!
3	4	2054702495	Sat Jun 06 07:54:59 PDT 2009	NO_QUERY	heya10	Getting ready to go skydiving. How exciting to...
4	0	2030659197	Thu Jun 04 08:41:44 PDT 2009	NO_QUERY	Sciteg	As i read a few minutes ago.. the visit of Ob...

feeler figure 3: sentiment140 dataset

2. **TextVectorization**: efficiently handles Twitter's noisy text, and the fixed-length sequences optimize Conv1D performance

Let vocabulary V contain top $N = 10,000$ tokens:

$$\mathbf{x}_i = \text{TextVectorization}(t_i) \in \mathbb{Z}^{100}$$

where t_i is a tweet, and $\mathbf{x}_i$ is its integer-encoded sequence (padded/truncated to length 100).

3. **Embedding Layer**

Projects token IDs into dense vectors:

$$\mathbf{E} \in \mathbb{R}^{N \times d} \quad (d = 128)$$

For input $\mathbf{x}_i$, the embedding output is:

$$\mathbf{X}_i = [\mathbf{E}_{x_{i1}}, \mathbf{E}_{x_{i2}}, ..., \mathbf{E}_{x_{i100}}] \in \mathbb{R}^{100 \times 128}$$

4. **Conv1D + GlobalPooling**: captures local n-gram patterns, more efficient than LSTM for short texts

Applies $K = 128$ filters with kernel size $k = 5$:

$$\mathbf{C}_i = \text{ReLU}(\mathbf{X}_i * \mathbf{W} + \mathbf{b})$$

where $\mathbf{W} \in \mathbb{R}^{5 \times 128 \times 128}$, $\mathbf{b} \in \mathbb{R}^{128}$, and $*$ denotes convolution.

Reduces sequence dimension:

$$\mathbf{p}_i = \text{MaxPool}(\mathbf{C}_i) \in \mathbb{R}^{128}$$

Final classification:

$$\mathbf{h} = \text{ReLU}(\mathbf{W}_1 \mathbf{p}_i + \mathbf{b}_1) \quad (\mathbf{W}_1 \in \mathbb{R}^{64 \times 128})$$
$$\hat{y}_i = \sigma(\mathbf{W}_2 \mathbf{h} + \mathbf{b}_2) \quad (\mathbf{W}_2 \in \mathbb{R}^{1 \times 64})$$

where σ is the sigmoid function.

5. **Dropout (0.4)**: prevents overfitting on imbalanced dataset, validation accuracy improved by 6.2%

6. **Loss Function and Optimizer**:

Binary cross-entropy over M samples:

$$\mathcal{L} = -\frac{1}{M} \sum_{i=1}^{M} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

Adam update rule with learning rate $\alpha = 0.001$:

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

where θ represents model parameters, $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected momentum estimates.

3.5 Data Pipeline Architecture/ Twitter-roBERTa-base E

Ingestion Layer:

- Connects to Twitter API v2 and Reddit API using PRAW (Python Reddit API Wrapper)

1. **Analysis Core**: the model, cardiffnlp/twitter-roberta-base-sentiment's reduces memory usage by 40% with <1% accuracy loss. Batch processing (32 samples/batch) optimizes GPU utilization
2. **Storage & Retrieval**: PostgreSQL database with optimized GIN index for text search
3. **Validation Approach**: tested against Stanford Sentiment Treebank (SST-2) and achieved 93.7% accuracy.

4 Experiments and Validation

Dataset Splitting and Performance Metrics: we did a 80/10/10 split for training , validation and testing respectively to see how the trained model generalized to problems during and after training. These are performance metrics we got:

Model	Accuracy	Precision	Recall	F1-Score
Sentient73	93.6%	0.88	0.91	0.89
Twitter-roBERTa	94.1%	0.93	0.95	0.94
Ensemble (Weighted)	92.7%	0.91	0.93	0.92

Sentient73 processed 5.8x more requests/minute on CPU

5 Results and Analysis

5.1 Presentation of Findings

feeler figure 4: sentient73 accuracy

We evaluate **Sentient73** using three key metrics: *accuracy*, *training stability*, and comparative *performance*.

Figure 4 shows the training/validation accuracy across 20 epochs. We can see from the figure the progression of either accuracy – the final values being 93.6% and 94.94% for the training and the validation respectively. The **≤2% gap** indicates no **overfitting**, validating our dropout (0.4) and early stopping design.

Wee have quantified the final test performance in table 3 below:

Metric	Sentient73	VADER (Baseline)	Improvement
Accuracy	93.6%	72.1%	+21.5 pp
F1-Score	0.93	0.71	+25.4%
Inference Speed (ms)	45	12	-73%

Table 3: performance metrics

NOTE: pp = percentage points. Trade-off: 3.8× slower than VADER but 24% more accurate.

5.2 Interpretation of Results

6 Discussion

6.1 Implication/Interpretation of Results

The described methodology presents a computationally efficient and transparent opportunity for consolidating and analyzing sentiment from NLP.

Our experimental validation demonstrates that **Feeler's hybrid architecture** achieves superior performance compared to traditional approaches:

Accuracy-Speed Tradeoff:
While **Sentient73** is 3.8× slower than VADER (Table 3), its 21.5 percentage point accuracy improvement justifies this tradeoff for business applications where precision is critical.

The ensemble approach (92.7% F1-score) proves that combining rule-based efficiency with **neural network** accuracy is viable.

Real-World Deployment:
In pilot testing, Feeler reduced manual analysis time by 78% while maintaining 89% agreement with **human annotators**.

The ≤2% train-val accuracy gap (Fig. 4) confirms the model's generalizability to unseen data.

6.2 Limitations of feeler

1. **Language Constraints:** currently limited to English text, though the architecture supports multilingual expansion.
2. **Sarcasm Detection:** misclassified 12% of sarcastic tweets (e.g., *"Just love when the app crashes!"* → Positive).
3. **Hardware Dependencies: roBERTa** component requires GPU for optimal performance (>5s latency on CPU).

6. 3 Comparative Advantage

Feeler outperforms existing solutions by combining:
Computational Efficiency: 45ms inference (vs. BERT's 850ms)
Business Adaptability: Modular design allows domain-specific fine-tuning (e.g., healthcare vs. retail)
Cost-Effectiveness: 1/10th the cloud hosting cost of commercial alternatives

Acknowledgements

The presented work has been supported by JHUB Africa. We would have thanked and list out many people that have contributed to this paper one way or the other, but for the limitation of space! Regardless, the authors gratefully acknowledge the support from them. Any opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily represent those of JHUB Africa or those that have directly or indirectly contributed.

7 Conclusion & Future Work

7.1 Key Contributions of our work

Technical Innovation: Hybrid CNN-Transformer architecture optimized for social media text**.** We Achieved 93.6% accuracy with <50MB model size

Practical Impact: Open-source Django platform deployable on-premise**.** And we have demonstrated 24% higher accuracy than VADER in African business contexts.

Research Value: Published training protocols for Sentiment140 adaptation**.** Benchmark dataset for emerging-market English dialects

7.2 Future Directions

- **Multilingual Expansion**: we consider integrating Swahili sentiment analysis using AfroXLMR later on when we are fairly comfortable with our work.
- **Edge Deployment**: Quantize models for mobile devices using TensorFlow Lite
- **Explainability**: add SHAP values to highlight sentiment-driving keywords
- **Vertical Specialization**: industry-specific modules (e.g., hospitality complaint triage), the potential field decreases in strength as see

REFERENCES

J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," NAACL, 2019.

V. Sanh et al., "DistilBERT, a Distilled Version of BERT," NeurIPS, 2020.

P. Sentiment140, "Twitter Sentiment Classification Dataset," 2009. [Online]. Available: http://help.sentiment140.com

A. Twitter-roBERTa, "TweetEval: Unified Benchmark for Twitter Sentiment Analysis," EMNLP, 2020.

Appendices

Appendix A: Sample Twitter-roBERTa Code

```
from transformers import pipeline
analyzer = pipeline("sentiment-analysis",
                    model="cardiffnlp/twitte
r-roberta-base-sentiment")
print(analyzer("The update ruined the user
experience!"))
# Output: [{'label': 'NEGATIVE', 'score':
0.98}]
```

Appendix B: Performance Metrics

Review Text	Prediction	Confidence
"Love this app!"	POSITIVE	0.99
"Meh, it's okay"	NEUTRAL	0.87
"Worst update ever"	NEGATIVE	0.96