# Building a Sentiment Classification Model for Fitness/Nutrition Applications' Reviews from Google US PlayStore

**Arnav Khinvasara**
akhinvasara@ucsd.edu

## 1 Introduction

Fitness and nutrition have become extremely critical over time, however, even with the rise of advanced technologies, it seems that there is no one perfect solution in terms of a handy application to help users pursue their goals. By classifying user sentiments in reviews of fitness apps, nutrition supplements, or diet plans, this could help companies understand customer satisfaction, identify common complaints, or improve their offerings based on feedback trends. Additionally, brand new features could be built upon existing infrastructure. By analyzing sentiment behind some user reviews we can help companies develop their products to possibly provide some personalized recommendations, identify trending topics, or improve overall.

**Project Goals**:

1. Explore sentiment analysis.
2. Pre-process/clean dataset.
3. Implement baseline/advanced models to classify sentiment.

It seems that all the major goals were accomplished. I was able to implement multiple models such as Logistic Regression, TextCNN, and DistilBERT, and the final model showed improved results compared to baseline results.

## 2 Related work

The study, "Motivations for user satisfaction of mobile fitness applications: An analysis of user experience based on online review comments" heavily influenced the overall scope of the project. In this study, the authors dove into how user satisfaction with fitness apps, including usability, functionality, and emotional design helped shed some light on patterns of satisfaction and dissatisfaction through users comments/reviews.

In fact, the dataset for this project was actually originated from this study as it not only contained online reviews of fitness apps, detailing user experiences but also gave a numerical value to several features representing the tone or mood of the user. This allowed me to look into class imbalance and other sentiments as there were more features available leading to more data points to use for the sentiment analysis.

Here were five relevant papers and how I had intended to make use from them for this sentiment analysis project:

1. **Aslam N, Ramay WY, Xia K et al. (2020) - Convolutional Neural Network Based Classification of App Reviews**: Explores the notion of using CNNs for classifying app reviews, which helped with the implementation of the TextCNN model.
2. **Hedegaard S, Simonsen JG (2013) - Extracting Usability and User Experience Information from Online User Reviews**: Explores extracting UX information from user reviews, which is helpful for preprocessing strategies.
3. **Dhiman N, Arora N, Dogra N et al. (2019) - Consumer Adoption of Smartphone Fitness Apps**: Discusses how consumer behavior has some correlation to fitness app adoption, which is relevant to the actual sentiment analysis.
4. **Huang YC, Chang LL, Yu CP et al. (2019) - Examining an Extended Technology Acceptance Model with Experience Construct on Hotel Consumers' Adoption of Mobile Applications**: Provides techniques

for analyzing user satisfaction, influencing review sentiment modeling.

5. **Sadiq S, Umer M, Ullah S et al. (2021) - Discrepancy Detection Between Actual User Reviews and Numeric Ratings of Google App Store Using Deep Learning**: Explores how the challenges of misaligned ratings/reviews can be combated - relevant for balancing ratings properly and textual sentiment in my pre-processing phase.

## 3 Dataset

The dataset consisted of **10,000 user reviews** and was collected reviews of the top five mobile fitness applications in the Google US Play Store as of two years ago. The dataset contains the textual reviews, LIWC results and star rating scores. Each review contained a user comment with a numerical star rating of 1–5, along with metadata such as the app name. Ratings were categorized as follows:

- **Positive**: Ratings 4–5
- **Neutral**: Ratings 2–3
- **Negative**: Ratings 1

**Dataset Statistics**

- **Total Reviews**: 10,000
- **Positive Reviews**: 70%
- **Neutral Reviews**: 20%
- **Negative Reviews**: 10%
- **Average Length**: 50 words per review
- **Vocabulary Size**: 20,000 unique words

**Issues with Dataset**

1. **Class Imbalance**: Positive reviews dominated dataset which led to challenges in model generalization for minority classes.

2. **Noise**: Reviews included typos, emojis, non-standard language, and other issues which required extensive pre-processing.

**Pre-processing Technique**

- **Text Cleaning**: Removed special characters, emojis, HTML tags, etc.
- **Lower-casing**: Standardized string to lowercase.
- **Stop-word Removal**: Excluded common words *(e.g. "and," "the").*
- **Tokenization**: Split text into tokens using a tokenizer.
- **Annotation**: Converted numerical ratings to sentiment labels.

**Examples**

| **Input** | **Sentiment** |
|---|---|
| *Amazing app! Helps me stay active.* | Positive |
| *It's okay, but the UI needs work.* | Neutral |
| *Terrible app, constant crashes!* | Negative |

## 4 Baselines

**Baseline Models**

1. **Logistic Regression**:
   - **Mechanism**: Models the relationship between word frequencies (TF-IDF features) and sentiment labels.
   - **Hyper-parameters**: Maximum iterations = 1000, Solver = lbfgs.
   - **Performance**: Accuracy: 84.6%; F1-Score: 85.4%.
2. **Naive Bayes**:
   - **Mechanism**: Assumes independence between features; uses word probabilities to predict sentiment.
   - **Hyper-parameters**: Laplace smoothing ($\alpha = 1$).
   - **Performance**: Accuracy: 83.3%; F1-Score: 83.4%.

**Data Splits**

To conduct proper data splits, I designed the Training Set to be 70%, the Validation Set to be 15%, and the Test Set to be 15%

**Methodology**: hyper-parameter tuning was not performed on the test set for proper, unbiased results.

Both the models provided decent enough baselines but then they struggled with class imbalance and it caused it to the perform poorly on neutral and negative sentiments.

## 5 My Approach

Instead of using such traditional models like Logistic Regression or Naive Bayes, I knew that I would have to leverage advanced neural networks capable of capturing complex relationships in text using models such as TextCNN or the lighter BERT: DistilBERT.

1. **TextCNN**: A convolutional neural network for text, capable of capturing n-gram features through convolutional filters. Traditionally, CNN's are used for images, however the same convolution practice can be efficiently applied for the sentiment analysis. It captures local dependencies within text through convolutional filters and then pools those outputs to portray global representations properly.
2. **DistilBERT**: A lightweight transformer-based model pre-trained on massive text corpora, however it is 40% lighter and more efficient than BERT, yet it is still able to capture 97% of BERT's accuracy.

**Working Implementation**

Here is a detailed explanation of the advanced model implementation.

**TextCNN Implementation**

**Key Components:**

- **GloVe Embeddings:** The model used **pre-trained GloVe embeddings (100-dimensional)** to represent words in the dataset. Although the pre-trained GloVe embeddings (200-dimensional) would provide better accuracy, it would be significantly more computationally exhaustive. The embeddings captured semantic relationships between words and the missing words were initialized to zero vectors.
- **Network Architecture:**
  - Embedding layer initialized with GloVe embeddings and fine-tuned during training.
  - Three convolutional filters with kernel sizes of **3, 4, and 5** were used to extract n-gram features.
  - Outputs from the convolutional layers were pooled, then concatenated, and then passed through a fully connected layer with a dropout at the end.
- **Training Process:**
  - Data tokenized and padded for equal sequence length, and represented as sequences.
  - The Weighted loss function (**CrossEntropyLoss**) was used to handle class imbalance.
  - The optimizer used was **AdamW**, combined with a **StepLR scheduler**.
- **Results:** After 10 epochs, the TextCNN model resulted in a **Test Accuracy of 88.3%**.

*Challenges:* Due to padding, there were longer training times in training. Also, regularization was significant to avoid over-fitting during hyperparameter fine-tuning.

**DistilBERT Implementation**

**Key Components:**

- **Tokenizer:** The **DistilBERT tokenizer** was used to tokenize and pad the text to a maximum sequence length of 128.
- **Model Architecture:**
  - Pre-trained **distilbert-base-uncased model** fine-tuned with a classification head.
  - Head mapped the DistilBERT embeddings to sentiment labels.
- **Training Process:**
  - Data was split into training/test sets (labels encoded into integers).
  - Optimized using **AdamW** with a learning rate of $5e^{-5}$.
- **Results:** After 10 epochs, the DistilBERT model achieved a **Test Accuracy of 87.0%**.

*Challenges:* The higher computational requirements increased training time for the larger datasets

**Compute Environment**

- **Device**: Apple Silicon MacBook Pro (M3 Pro Chip) with ARM64 architecture
- **Frameworks**: PyTorch, Hugging Face Transformers

Both models ran locally; didn't require cloud GPU use.

**Runtime**

- **TextCNN**: 10 epochs, approximately 2 minutes per epoch.
- **DistilBERT**: 10 epochs, approximately 10 minutes per epoch.

**Results**

| **Model** | **Metrics (%)** | | | |
|---|---|---|---|---|
| (lr)2-5 | Acc. | Prec. | Rec. | F1 |
| Logistic Regression | 84.6 | 85.0 | 87.4 | 85.4 |
| Naive Bayes | 83.3 | 83.3 | 87.0 | 83.4 |
| TextCNN | 88.2 | 86.5 | 88.3 | 87.5 |
| **DistilBERT** | **87.0** | **88.0** | **87.0** | **88.0** |

Table 1: Model Performance Comparison

**Detailed Analysis of DistilBERT**

- **Training Progress**:
  - Initial accuracy: 87.7% in epoch 1.
  - Final accuracy: 98.3% on training data by epoch 10.
- **Classification Performance**:
  - **Positive Sentiment**: Excellent performance (96% precision and 95% recall).
  - **Neutral Sentiment**: Improved slightly but remained challenging (44% recall).
  - **Negative Sentiment**: Outperformed TextCNN in minority class recognition (62% recall)

**Key Takeaways**

1. **Strengths**: DistilBERT consistently outperformed TextCNN in handling the more nuanced sentiments which showed it benefited from its ability to consider context.
2. **Challenges**: Both models struggled with the minority classes (neutral and negative).

## 6 Error Analysis

**Baseline Model Failures**

The baseline models, Logistic Regression and Naive Bayes, struggled significantly with inputs containing:

- **Ambiguous Sentiments:** Sentences with mixed positive and negative phrases often led to incorrect classifications. For example:
  - Input: "The app has great features, but the UI is frustratingly slow."
  - Predicted: Positive
  - Actual: Neutral
- **Uncommon Words/Slang:** Sentences with rare vocabulary and informal expressions were not well-represented in the training data so it often resulted in misclassification. If there was a library that was able to determine slang sentiment, it would be heavily beneficial.
  - Input: "Totally vibing with this app!"
  - Predicted: Neutral
  - Actual: Positive
- **Short Sentences:** Very short reviews such as "Not great" were often misclassified as Neutral instead of Negative.

**Advanced Approach Failures**

**TextCNN and DistilBERT** showed significantly improved performance but there will still many challenges:

- **Complex Syntax:** Sentences with nested clauses or complex grammar and intricate words.
  - Input: "While the app is functional, its bugs make it barely usable."
  - Predicted: Neutral
  - Actual: Negative
- **Subtle Sentiments:** Reviews with subtle emotional cues or sarcasm often led to errors.
  - Input: "Well, I guess it works if you're okay with waiting forever."
  - Predicted: Neutral
  - Actual: Negative

- **Minority Classes:** Even with class weight adjustments, Neutral/Negative classes had slightly lower precision/recall which exhibits class imbalance.

**Semantic/Syntactic Commonalities in Errors**

- Inputs with **mixed sentiments** (e.g., positive and negative phrases in the same sentence) seemed to cause confusion across all four models.
- **Sarcasm and implicit negativity** was not at all detected by the models. Considering that sarcasm can sometimes be hard even for humans to detect on text, maybe more advanced models might be able to detect sarcasm.
- Reviews with **domain-specific lingo** were problematic due to insufficient representation in training data.

**Observations and Hypotheses**

- Context was very important and we could see this through how error-prone the model was as they relied on simple word frequency statistics.
- Even though TextCNN captured some context, it struggled with highly complex syntax because long-range dependencies are hard to model.
- Ultimately, DistilBERT was the better performer overall but it showed weaknesses in handling sarcasm as well as extreme class imbalance.

## 7 Conclusion

A large key takeaway with this sentiment analysis project was the importance of designing models to address the underrepresented classes, as this significantly influenced the overall performance and fairness across sentiment categories. Positive reviews were the clear majority which is why detecting positive sentiment was far more precise, however, detecting negative/neutral sentiment accurately was a challenge because of how little training data there was. Luckily, the integration of pre-trained embeddings like GloVe and advanced models like DistilBERT provided clear improvements over traditional baselines.

Another key aspect of this project was how critical of a role that both hyperparameter tuning and dataset preprocessing played such as weighting the loss function and making fine-tune adjustments. The results before and after showed just this. Similar to the practice of ablation, the absence of some of these hyperparameters proved to be necessary.

Looking ahead, just like the teaching assistant had advised, exploring zero- or few-shot prompting could offer a compelling baseline for comparison. Tools like `llama.cpp` for local inference could helped with even specialized domains such as ”nutrition” or ”fitness”. I think the best way would be to find more data and reviews that are negative/neutral since the class imbalance is what is stopping the current model from being even more accurate in a more significant manner. Expanding the dataset would be ideal.

## 8 Acknowledgements


- **Sections of the Report:** ChatGPT was used to help create functions to evaluate my logistic regression, naive bayes, TextCNN, and DistilBERT models. It also assisted in debugging and logging during implementation. Additionally, ChatGPT provided guidance on structuring the final report and generating tables and some other information in LaTeX.
- **Code Modifications:** Some code was reused and modified from CSE256 Programming Assignments 1 and 2.
- **Degree of Changes:** Many minor changes were used in code, almost none was used for the writing portion of the project.


References:


- (Ahn and Park, 2023)
- (Aslam et al., 2020)
- (Dhiman et al., 2020)
- (Huang et al., 2019)
- (Sadiq et al., 2021)
- (Hedegaard and Simonsen, 2013)
- CSE256 Programming Assignment 1
- CSE256 Programming Assignment 2

## References


Ahn, H. and Park, E. (2023). Motivations for user satisfaction of mobile fitness applications: An analysis of user experience based on online review comments. *Humanities and Social Sciences Communications*, 10(1):1–7.

Aslam, N., Ramay, W. Y., Xia, K., and Sarwar, N. (2020). Convolutional neural network based classification of app reviews. *IEEE Access*, 8:185619–185628.

Dhiman, N., Arora, N., Dogra, N., and Gupta, A. (2020). Consumer adoption of smartphone fitness apps: an extended utaut2 perspective. *Journal of Indian Business Research*, 12(3):363–388.

Hedegaard, S. and Simonsen, J. G. (2013). Extracting usability and user experience information from online user reviews. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, page 2089–2098, New York, NY, USA. Association for Computing Machinery.

Huang, Y.-C., Chang, L. L., Yu, C.-P., and Chen, J. (2019). Examining an extended technology acceptance model with experience construct on hotel consumers' adoption of mobile applications. *Journal of Hospitality Marketing & Management*, 28(8):957–980.

Sadiq, S., Umer, M., Ullah, S., Mirjalili, S., Rupapara, V., and Nappi, M. (2021). Discrepancy detection between actual user reviews and numeric ratings of google app store using deep learning. *Expert Systems with Applications*, 181:115111.