Simple Review - cs510 Final Project Report

Jiayi Gu
University of Illinois Urbana-Champaign
Champaign, IL, USA
jiayigu4@illinois.edu

Wentao Zhou
University of Illinois Urbana-Champaign
Champaign, IL, USA
wentao6@illinois.edu

Kanghong Zhao
University of Illinois Urbana-Champaign
Champaign, IL, USA
kz8@illinois.edu

Xiaoqing Yao
University of Illinois Urbana-Champaign
Champaign, IL, USA
xyao14@illinois.edu

ABSTRACT

The rapid growth of e-commerce has magnified the role of online product reviews in consumer purchasing decisions. However, it can be difficult to extract useful information and keywords from the vast amount of reviews. Our project, Simple Review, addresses this issue by developing a tool to summarize key aspects of product reviews and enable consumers to make decisions based on the emphasized key aspects efficiently. By utilizing advanced natural language processing techniques, our tool extracts positive and negative sentimental/key aspects within reviews and highlights the strengths and weaknesses of products. We aim to simplify the whole process of reading and selecting the most desirable product for the consumers.

ACM Reference Format:
Jiayi Gu, Wentao Zhou, Kanghong Zhao, and Xiaoqing Yao. 2024. Simple Review - cs510 Final Project Report. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In the digital age, online shopping has become a predominant mode of commerce, making product reviews crucial in guiding consumer decisions. Recognizing the influential power of reviews, our project, Simple Review, aims to enhance the shopping experience by streamlining the review analysis process. Consumers all tend to choose products with good ratings and steer clear of those with bad ones. However, it can be a time-consuming and overwhelming task for them to sift through extensive reviews to discern product quality. Our tool addresses this challenge by succinctly summarizing the essence of reviews, focusing on the key positives and negatives conveyed by customers.

The motivation behind Simple Review is rooted in the observation that while consumers rely heavily on product ratings and reviews, the process of reading through these reviews is cumbersome and can deter potential purchases. Our solution leverages advanced linguistic models to extract meaningful insights from text, allowing users to quickly grasp what other consumers appreciate or dislike about a product. This not only saves time but also enhances the decision-making process by highlighting the most related keyword and information.

2 ARCHITECTURE

2.1 Overview

The architecture of Simple Review is designed to be robust yet straightforward, ensuring efficient handling of data and interactions across different components of the system. The core components include the frontend user interface, backend server, database, and keyword extraction model. Each component is optimized for its role within the system, from user interaction to data processing.

2.2 Frontend

The frontend is developed using HTML and CSS, providing a clean and intuitive user interface that displays products and their associated keywords. This simplicity ensures that users can easily navigate and interact with the system without being overwhelmed by unnecessary complexity. The website is shown in Figure1

2.3 Backend

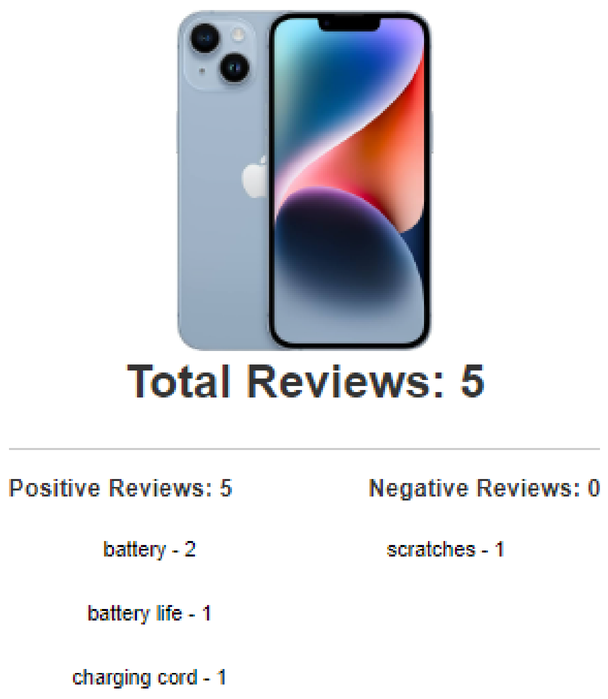
The backend is built with Python using the Flask framework, known for its simplicity and effectiveness in handling web server tasks. Flask serves as the intermediary between the frontend and the database, processing user requests, and executing the appropriate actions, such as fetching data from the database or initiating keyword extraction processes.

2.4 Database

We chose MongoDB for our database needs due to its flexibility and performance with large datasets, which is ideal for handling extensive product reviews. MongoDB stores all reviews and the extracted keywords, supporting fast retrieval which is critical for providing a responsive user experience.

2.5 Models

Our system incorporates two main models for keyword extraction: Traditional NLP Tools: This approach uses a combination of BART for summarizing reviews, NLTK's SentimentIntensityAnalyzer for sentiment analysis, and spaCy for keyword extraction. Despite


Total Reviews: 5

Positive Reviews: 5	Negative Reviews: 0
battery - 2	scratches - 1
battery life - 1	
charging cord - 1	

Figure 1: Demo website

its utility, this model sometimes struggles with producing precise keywords, prompting us to refine our methodology. LLM-Based Model (Gemini): Following our experience with Gemini in previous assignments, we adapted it to directly handle entire reviews for keyword extraction. This model, while not preserving state across different prompts, provides better context understanding and has shown promising results in extracting relevant keywords.

2.6 Integration and Workflow

The entire system is designed to facilitate a smooth workflow. When a user queries a product, the backend retrieves relevant reviews from the database, processes them through the selected model to extract keywords and sentiments, and then sends this processed information back to the frontend for display. This workflow is optimized for speed and accuracy, ensuring that users receive timely and relevant information. Figure 1 shows the overall architecture for our project.

3 IMPLEMENTATION DETAILS

3.1 Traditional NLP

The traditional NLP approach consists of four parts: Summarization, Sentence Splitting, Sentiment Detection, and Keywords Extraction. Each part had a thorough comparison of various toolkits and models and the optimal one was selected for our project.

Within a review, there can be many unnecessary parts that are not related to the product itself. For example, discussing a terrible experience with another product to showcase how much one likes the current product will not provide any information about the features of the product. These unrelated opinions can sometimes confuse the extraction model, leading to incorrect keywords. In order to solve this issue, we decided to summarize the reviews first to make them concise and filter out the unimportant parts. After meticulous research, we found three well-known NLP models: BART[6], T5[1], and Pegasus[5]. These models all had great performance in summarizing long paragraphs. However, for our project, most reviews are quite short with only a few sentences. Therefore, it was challenging to determine which model is the best for summarizing such short reviews. In this case, we referred to another research paper titled "An Analysis of Abstractive Text Summarization Using Pre-trained Models."[7] This paper specifically compared the performance of these three NLP models across three different datasets. Among these datasets, BART exhibited an overall better performance. Hence, we decided to proceed with BART for the review summarization of our project.

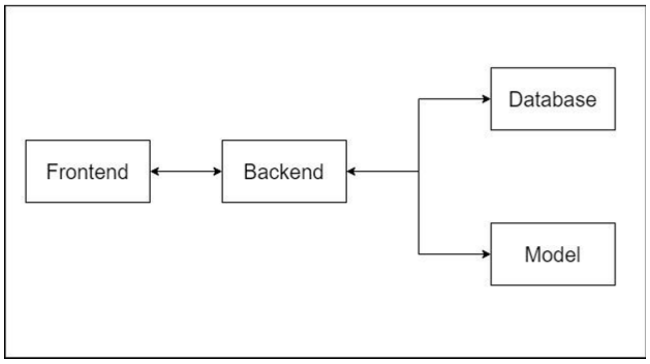


Figure 2: figure to show overall architecture.

For Sentence Splitting, we simply used regular expression matching to segment the review into multiple sentences. This step is crucial as our project aims to display both positive and negative keywords. Even in a positive review, there might be mentions of negative aspects of the product that we need to extract. In this case, we split the review and conducted sentiment analysis on each sentence to determine whether it discusses the positive or negative aspects of the product.

As mentioned earlier, conducting sentiment analysis on each sentence of the review is crucial for classifying keywords as positive or negative. In order to do so, we initially tested a well-known open-source NLP toolkit called Textblob. It successfully identified the sentiment of the review; however, the generated scores are often close to the boundary of positive or negative, even for reviews with obvious sentiment. Therefore, we decided to explore another toolkit capable of detecting sentiment more confidently. After further research, we found another toolkit called NLTK with VADER-Sentiment-Analysis[4]. Even though this toolkit is a lot older than Textblob, it is specifically tuned to identify sentiments expressed in social media. This aligns perfectly with our needs since reviews can be considered a form of social media content. After some testing, VADER proved its strength by providing high confidence in correct sentiment classification.

The key component of the traditional NLP approach is keyword extraction since it significantly impacts the model's performance. Initially, we planned to utilize the Latent Dirichlet Allocation (LDA)[3] model to extract topics. However, we realized that LDA requires a collection of documents to extract topics from each one, which doesn't align with our project's requirement of receiving and processing one sentence at a time. Because of this, we need to find some other toolkits to extract keywords from a single sentence. Later, we found some other toolkits like Rake, Yake, and Keybert. Among these, Rake and Yake often produced unusual phrases that weren't suitable for our needs. Keybert showed promise but it had another problem. Its output was a dictionary of keywords with scores and it included various types of words other than nouns. Because of this, the scores of noun words might be overshadowed which make it difficult to set a threshold for determining keywords. To address this, we decided to extract all noun words as keywords. This had similar results as Keybert but with more actual keywords extracted. The final toolkit we used to extract all noun words is spaCy.

3.2 LLM

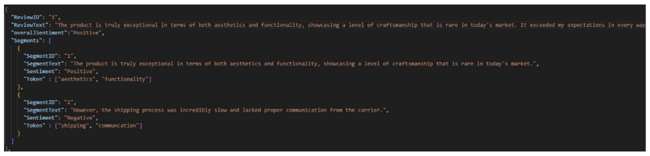
Since Large Language Models (LLMs) are currently a trending topic, we also wanted to test their performance for our project. While there are various LLMs like GPT available, most of them do not offer a free API. The only free API we found is Gemini[2] which is a multimodal large language model developed by Google. Initially, we also applied the summarization and splitting techniques we used for the traditional model to Gemini. However, the summarization output of Gemini is uncontrollable. It tends to convert verbs or adjectives to nouns based on its understanding even if we told it not to. This is not a problem for the application itself, but this is an issue for evaluation as we must keep it consistent by restricting words to nouns only. Unfortunately, this approach didn't work as we expected most of the time. Regarding splitting, a problem with Gemini API is that it treats each prompt independently. If a review compares one product with another in separate sentences, Gemini may focus on the other product since it lacks context about which entity is being referred to. Because of these two problems, the performance of Gemini is far below our expectations. As a solution, we decided to send the entire review to Gemini and continuously fine-tune the prompt to improve the results.

4 USER GUIDE

The user guide provides step-by-step instructions on how to set up and interact with our review analysis tool. This guide aims to assist users from installation to the practical use of the application, ensuring they can leverage the tool's capabilities effectively.

4.1 System Requirements and Installation

The application requires Python 3.6 or later and access to a MongoDB server, which manages the storage and retrieval of review data. Users must first clone the project repository using Git and then navigate to the project directory. Here, a virtual environment should be created and activated to manage dependencies without affecting system-wide Python settings. Dependencies are installed using pip, Python's package installer, from the provided `requirements.txt` file. This setup ensures that the application environment is correctly configured without compatibility issues.



```
{
  "ReviewID": "1",
  "ReviewText": "The product is truly exceptional in terms of both aesthetics and functionality, showcasing a level of craftsmanship that is rare in today's market. It exceeded my expectations in every way
  "overallSentiment":"Positive",
  "Segments": [
    {
      "SegmentID": "1",
      "SegmentText": "The product is truly exceptional in terms of both aesthetics and functionality, showcasing a level of craftsmanship that is rare in today's market.",
      "Sentiment": "Positive",
      "Token" : ["aesthetics", "functionality"]
    },
    {
      "SegmentID": "2",
      "SegmentText": "However, the shipping process was incredibly slow and lacked proper communication from the carrier.",
      "Sentiment": "Negative",
      "Token" : ["shipping", "communcation"]
    }
  ]
},
```

Figure 3: Manually label the review into different sentiments with keywords

4.2 Launching the Application

The application is initiated by running the `app.py` file within the backend directory. Upon successful launch, users can access the application by navigating to http://127.0.0.1:5000/ on any web browser. This simplicity ensures that users without technical expertise can also interact with the system with ease.

4.3 Operational Guide

Users interact with the application through a simple web interface where they can submit product reviews. Upon submission, the system employs the Gemini model to analyze the text and extract significant positive and negative keywords. These keywords are then displayed under the 'Positive Reviews Keywords' and 'Negative Reviews Keywords' sections on the homepage. The interface updates to show the top ten keywords for each category based on frequency, aiding users in quickly understanding the prevailing sentiments in product reviews.

5 EVALUATION

5.1 Evaluation Dataset

To evaluate the performance of our keyword extraction models, we utilized the Amazon Reviews 2023 dataset from an open-source GitHub repository. We selected 100 reviews from this dataset and manually labeled the keywords for each review, as we could not find an existing dataset with the required information. This manually labeled dataset served as the ground truth for comparing the generated keywords from our traditional and LLM models as shown in Fig. 2.

5.2 Precision, Recall, F1-score

We calculated the precision, recall, and F1-score of the extracted keywords for both the traditional and LLM models. Precision is the proportion of correctly extracted keywords out of all extracted keywords. The recall is the proportion of correctly extracted keywords out of all labelled keywords in the ground truth. F1-score is the harmonic mean of precision and recall.

One evaluation technique we applied is loose matching. Since a word can have different forms with the same meaning, a strict match might miss such cases and result in failed matches which will significantly impacts the performance of the model. To implement loose matching, we utilized the similarity function from the spaCy NLP toolkit to calculate the similarity score between two words.

	Precision	Recall	F1-score
Positive Keywords	0.30	0.45	0.36
Negative Keywords	0.33	0.46	0.38
Overall	0.31	0.46	0.37

Table 1: Traditional Model

	Precision	Recall	F1-score
Positive Keywords	0.57	0.77	0.66
Negative Keywords	0.56	0.75	0.64
Overall	0.57	0.76	0.65

Table 2: LLM Gemini

Then we manually set a threshold and consider any two words with a similarity score beyond the threshold as a match.

The evaluation results are shown in Table 1 and Table 2. The performance of the traditional model, which used NLTK, BART and spaCy, was quite poor. It generated many more keywords than the actual keywords, leading to low precision, recall and F1-score across both positive and negative keywords and overall.

In contrast, the LLM model using Google Gemini achieved significantly higher recall, although its precision and F1-score were not as high. For our project, recall is more important than precision, as a high recall indicates the model's ability to capture most of the actual keywords. The extra keywords generated by the model can be filtered out by the ranking system that displays the top 10 most frequent keywords. With an overall recall of 0.76, the Gemini model performs quite well for our use case.

6 FUTURE WORK

The roadmap for future development focuses on enhancing the tool's capabilities through advanced modelling techniques, interface improvements, and expanding accessibility. These initiatives aim to not only refine the user experience but also to extend the tool's utility and security as it scales.

6.1 Advanced Model Integration

Future enhancements will consider integrating more sophisticated language learning models such as GPT-4 or OpenAI Codex. These models offer improved accuracy in sentiment analysis and keyword extraction by understanding the contextual nuances within the reviews better than the current models.

6.2 User Interface Improvements

The user interface of the application, currently utilitarian, is planned to be developed into a more engaging and visually appealing format. Additional functionalities such as real-time keyword updates, interactive sentiment analysis graphs, and a dynamic review presentation could significantly enhance user engagement and usability.

6.3 Database and Security Enhancements

As the application scales, a more robust database infrastructure will be necessary to handle an increasing volume of data efficiently. Security is another critical area, with plans to implement state-of-the-art security measures to protect user data, ensuring privacy and data integrity as the user base expands.

6.4 Mobile Compatibility and Accessibility

Considering the increasing use of mobile devices for accessing applications, a mobile-responsive version of the application will be developed. This version will cater to users on-the-go and is expected to broaden the application's user base significantly.

7 SUMMARY

This report details the development and evaluation of Simple Review, a tool designed to enhance the online shopping experience for our consumers by providing quick and clear summaries of product reviews. Our project leverages advanced natural language processing techniques to extract and analyze key sentiments and phrases from extensive reviews, which helps potential customers avoid reading an overwhelming amount of review texts to get the information they want. The system's architecture consists of a user-friendly frontend, a robust Python flask backend, and a MongoDB database, along with two keyword extraction models(a traditional NLP model and a more dynamic model- Gemini). Our comprehensive dataset is meticulously prepared and well-structured for our model. The evaluation metrics employed include F1-score, recall, precision and confusion matrix, providing a detailed assessment of each model's effectiveness. It also reveals areas of potential future enhancement, including incorporating more sophisticated models, enhancing user interface and extending the application's reach across different platforms.

8 CONTRIBUTION

- Wentao Zhou: Researched various models and implemented the desired ones for the two approaches we had for analyzing the reviews.
- Jiayi Gu: responsible for designing and implementing the database and back-end components of the Simple Review project. Also, established the connection between the frontend, backend, and database, ensuring seamless data flow and efficient system integration.
- Kanghong Zhao: Prepared and manually labeled a diverse dataset that is structured to facilitate precise model training. Finished the evaluation part of our keyword extraction model.
- Xiaoqing Yao: responsible for front-end development and setting up the project development environment.

REFERENCES

[1] Adam Roberts Katherine Lee Sharan Narang Michael Matena Yanqi Zhou Wei Li Peter J. Liu Colin Raffel, Noam Shazeer. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv:1910.10683* (2019).
[2] Gemini Team Google. 2023. Gemini: A Family of Highly Capable Multimodal Models. *arXiv:2312.11805* (2023).
[3] Chi Yuan Xia Feng Xiahui Jiang Yanchao Li Liang Zhao Hamed Jelodar, Yongli Wang. 2017. Latent Dirichlet Allocation (LDA) and Topic modeling: models, applications, a survey. *arXiv:1711.04305* (2017).
[4] C.J. Hutto and E.E. Gilbert. 2014. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. In *Proceedings of the Eighth International Conference on Weblogs and Social Media (ICWSM-14).*

[5] Mohammad Saleh Peter J. Liu Jingqing Zhang, Yao Zhao. 2019. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. *arXiv:1912.08777* (2019).
[6] Naman Goyal Marjan Ghazvininejad Abdelrahman Mohamed Omer Levy Ves Stoyanov Luke Zettlemoyer Mike Lewis, Yinhan Liu. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv:1910.13461* (2019).
[7] Debarshi Kumar Sanyal Samiran Chattopadhyay Tohida Rehman, Suchandan Das. 2023. An Analysis of Abstractive Text Summarization Using Pre-trained Models. *arXiv:2303.12796* (2023).