Application Info Extractor Report

1 Introduction

This report details a web-based sentiment analysis application built using Flask, a Python web framework. The application allows users to input a search query, fetches results from Google using SerpAPI, and performs sentiment analysis on the results to categorize them as good or bad reviews. The system leverages caching for efficiency and provides a user-friendly interface to display the results.

2 System Overview

The application integrates several components to perform sentiment analysis on web search results:

- **Flask Framework**: Handles web routing, templating, and user interaction.
- **SerpAPI**: Fetches search results from Google based on user queries.
- **Cachetools TTLCache**: Caches search results for 1 hour to reduce redundant API calls.
- **Sentiment Analysis**: Analyzes search result snippets for positive and negative keywords to determine sentiment.
- **Web Interface**: Provides a form for query input and displays results with sentiment metrics.

3 Implementation Details

3.1 Dependencies

The system relies on the following Python libraries:

- flask: For building the web application.
- cachetools: For caching search results.
- serpapi: For querying Google search results.
- requests: For handling HTTP requests (used internally by SerpAPI).
- asyncio: For asynchronous API calls.

3.2 Key Features

The application includes the following features:

- **Search Query Handling**: Users input a query and the number of results via a web form.
- **Caching**: Uses TTLCache to store results for 1 hour, reducing API calls.
- **Sentiment Analysis**: Counts good and bad reviews based on predefined keyword lists (e.g., "bad," "good," "terrible," "excellent").
- **Error Handling**: Manages timeouts and API errors gracefully.
- **Result Display**: Shows the number of good and bad reviews, a sentiment percentage, a sentiment category (e.g., "Mostly Good"), and the URLs of the search results.

3.3 Processing Pipeline

The system processes user requests as follows:

1. **User Input**: The user submits a query and the desired number of results via a web form.
2. **Cache Check**: Checks if the query results are cached; if so, returns them.
3. **API Call**: If not cached, uses SerpAPI to fetch search results asynchronously.
4. **Sentiment Analysis**: Analyzes the title, snippet, and URL of each result for positive and negative keywords.
5. **Result Compilation**: Calculates the number of good and bad reviews, sentiment percentage, and category.
6. **Output Rendering**: Renders the results in an HTML template for display.

4 Sample Output

During testing with the query "Python programming reviews" and a request for 10 results, the system might produce the following output:

- **Query**: Python programming reviews
- **Number of Good Reviews**: 7
- **Number of Bad Reviews**: 2
- **Sentiment Percentage**: 77.8% (calculated as 7 / (7 + 2) * 100)
- **Sentiment Category**: Good (since 77.8% is between 60% and 80%)
- **Search Result URLs**:

  - https://www.example.com/python-reviews (snippet: "Python is an excellent language for beginners.")
  - https://www.example.org/python-issues (snippet: "Some users report terrible performance with Python.")
  - https://www.pythonreviews.com (snippet: "A fantastic tool for developers.")
  - (Additional URLs omitted for brevity)

The web interface would display this information in a structured format, with the sentiment metrics prominently shown and the URLs listed as clickable links.

5 Performance Optimizations

The system includes optimizations to improve performance:

- **Caching**: Reduces API calls by caching results for 1 hour.
- **Asynchronous API Calls**: Uses asyncio to handle API requests efficiently.
- **Limited Result Processing**: Processes only the requested number of results to minimize computation.

6 Challenges and Limitations

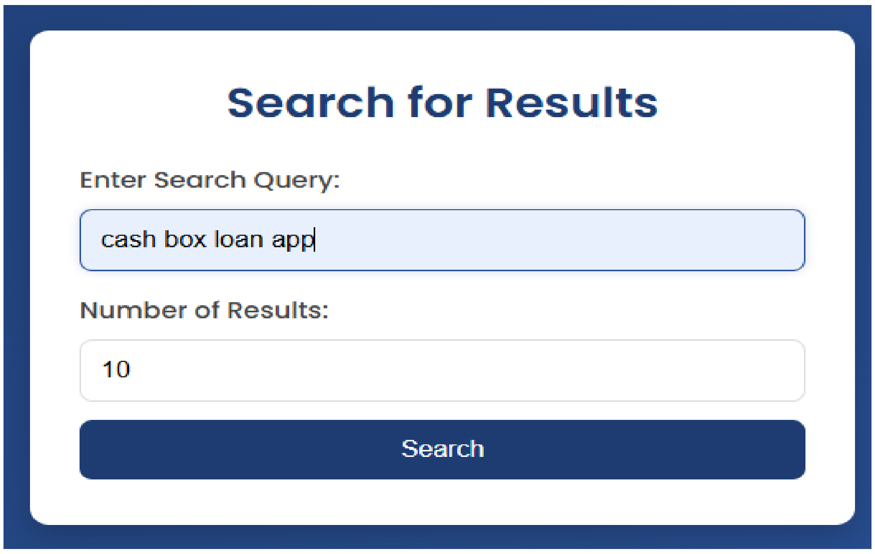
- **Sentiment Accuracy**: Relies on keyword matching, which may miss nuanced sentiment (e.g., sarcasm or context-dependent phrases).
- **API Dependency**: Requires a valid SerpAPI key and is subject to API rate limits or downtime.
- **Result Relevance**: SerpAPI results may not always be relevant for sentiment analysis (e.g., news articles vs. reviews).
- **Keyword Limitations**: The predefined keyword lists may not capture all sentiment variations.

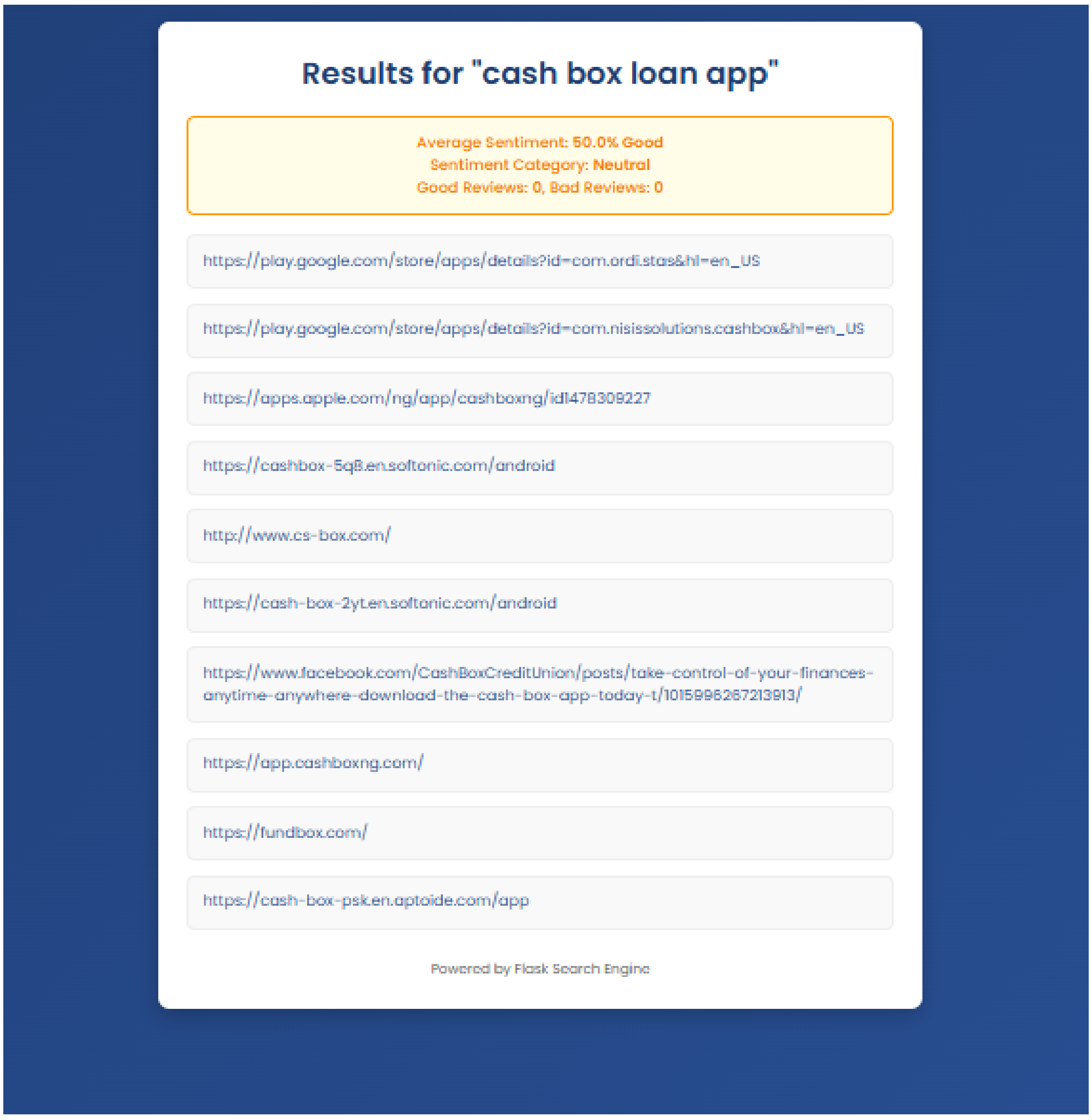
7 Future Improvements

Potential enhancements include:

- **Advanced Sentiment Analysis**: Use natural language processing (NLP) models like BERT for more accurate sentiment detection.
- **User Customization**: Allow users to define their own sentiment keywords.
- **Result Filtering**: Filter results to focus on review-specific sources (e.g., exclude news articles).
- **Visualizations**: Add charts or graphs to visualize sentiment trends over multiple queries.

8 Output:





Results for "cash box loan app"

Average Sentiment: 50.0% Good
Sentiment Category: Neutral
Good Reviews: 0, Bad Reviews: 0

https://play.google.com/store/apps/details?id=com.ordi.stas&hl=en_US

https://play.google.com/store/apps/details?id=com.nisissolutions.cashbox&hl=en_US

https://apps.apple.com/ng/app/cashboxng/id1478309227

https://cashbox-5q8.en.softonic.com/android

http://www.cs-box.com/

https://cash-box-2yt.en.softonic.com/android

https://www.facebook.com/CashBoxCreditUnion/posts/take-control-of-your-finances-anytime-anywhere-download-the-cash-box-app-today-t/1015996267213913/

https://app.cashboxng.com/

https://fundbox.com/

https://cash-box-psk.en.aptoide.com/app

Powered by Flask Search Engine

9 Conclusion

The sentiment analysis web application provides a simple and effective way to gauge the sentiment of online content based on user queries. By integrating SerpAPI, caching, and a Flask-based interface, it delivers quick insights into the proportion of good and bad reviews. While the keyword-based sentiment analysis has limitations, the system offers a solid foundation for further enhancements, such as advanced NLP techniques and improved result filtering.