COMPA: A COMPARATIVE RETRIEVAL AND ANALYTICAL ENGINE FOR CONSUMER PRODUCTS

by

Ramsés Jonathan Sotelo Jiménez, B.S.

A thesis submitted to the Graduate College of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Computer Science
May 2022

Committee Members:

Byron J. Gao, Chair

Anne H.H. Ngu

Xiao Chen

FAIR USE AND AUTHOR'S PERMISSION STATEMENT

Fair Use

Duplication Permission

ACKNOWLEDGEMENTS

I would like to acknowledge my advisor, Dr. Byron Gao for all his support throughout this thesis. I would also like to thank my committee members for agreeing to be on my committee. Thank you to Bhavya Medishetty for helping me with my research and refining my algorithm. I would also like to thank one of my mentors, Swapna Valluri for all her support while I worked on this Thesis. Thanks to my buds Ryan, Luke and Tim for all your support while I have been working on my Thesis I appreciate it. Major thanks to my parents(Ramses and Socorro), my brother Sarek, my sister Katherine and my fiancé Vanessa Avila for all their love and support throughout this process, I know it's been tough, but this is finally done.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

ABSTRACT

Often people need to make comparisons in order to make selections and decisions. Increasingly such comparison activities are conducted online. For example, consumers typically read many reviews and compare various models before an online purchase. Since this comparison-based analytical and decision-making process often involves significant manual effort, it is greatly beneficial if the process can be largely automated and effectively performed by computers. In this thesis, we make an effort to design and implement COMPA, a web-based comparative retrieval and analytical engine in the context of consumer products, which takes two entities as input and returns a comprehensive comparison report to effectively assist decision-making.

I. INTRODUCTION

Comparative assessment of options is crucial in decision-making especially in E-commerce, where consumers typically read many reviews and compare various models before an online purchase. This comparison-based analytical and decision-making process can be very time-consuming involving significant manual effort, which motivates our study to design and implement COMPA, a comparative retrieval and analytical engine that aims to effectively automate this process and assist decision-making.

COMPA is a web-based system implemented in the context of consumer products, however, the idea and system can be directly applied to other application domains involving online comparative decision-making. It features a simple web interface that takes user-input queries similar to web search. For a given query in the form of A vs B, where A and B are two entities (e.g., consumer product models), COMPA will search through various data sources (e.g, Amazon product reviews) and generate a comprehensive comparison report that includes comparison scores as well as supporting evidence.

As far as we know COMPA is one of the first comparative analytical systems of its kind. COMPA is related to comparative opinion mining, a sub-field of opinion mining. *Opinion mining*, or *sentiment analysis* is "the field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities" [1]. *Comparative opinion mining* can be referred to as the task of extracting valuable comparative opinions from text. Relatively little research has been done in comparative opinion mining as it is a relatively new line of research.

To generate a comparison report for the A vs B query, COMPA performs comprehensive analysis that integrates three components: independent analysis, comparative analysis, and inference analysis. Independent analysis is where we

analyze reviews/comments containing only A or only B. Comparative analysis is where we perform a direct comparison analysis for A and B based on comparative reviews involving both A and B. With comparative analysis, we are able to calculate how A and B directly compare. Inference analysis is where we don't have a direct comparison between A and B but instead both A and B have a comparison with a third entity C, so we can infer how A and B compare by how A and C compare and B and C compare. The produced comparison report contains 'Relative Strength' scores that suggest how A and B compare in the eyes of consumers. For example, 0.7 vs 0.3 suggests 70% of consumers prefer product A whereas 30% prefer product B. The comparison report also contains evidence supporting these scores for the user to verify. The supporting evidence contains highlighted keywords from product reviews. By verifying the evidence, the user can assign a degree of confidence for the calculated comparison scores.

Contributions

This thesis work makes the following contributions.

- We propose a novel comparative retrieval and analytical engine in the context of consumer products that potentially has great practical significance.
- Our system, COMPA, is designed to perform comprehensive analysis by integrating three components: independent analysis, comparative analysis, and inference analysis.
- We implement COMPA as a publicly accessible system running on real data sets, demonstrating the utility of the system.
- We collect and process multiple real consumer product review data sets that can be shared with other researchers for the study of comparison opinion mining.

- The COMPA system and methodology can be easily adapted to other application domains involving online comparative decision-making.

This thesis is divided into 6 chapters, Chapter 2 will go over a background on opinion mining and some related work in the field. Chapter 3 will discuss our methodology before we go into our implementation of COMPA in Chapter 4. Chapter 5 will provide a demonstration of COMPA and some experiments using COMPA. Chapter 6 will offer final thoughts as well as discuss future work that can be done with COMPA.

II. RELATED WORK

Sentiment Analysis

Sentiment analysis techniques can be broadly classified into three categories Rule based, Machine Learning based, and Natural Language Processing [2]. In COMPA we are primarily focusing on libraries that use Rule based and Natural Language Processing otherwise known as NLP. Current state-of-the-art sentiment analysis techniques use concepts of NLP and Deep learning. In recent years, Deep Learning models have been extensively applied in the field of NLP and show great potential [3]. Recently substantial work has shown that pre-trained models (PTMs), on the large corpus can learn universal language representations, which are beneficial for downstream NLP tasks and can avoid training a new model from scratch [4]. These methods might not be perfect for analyzing product reviews because it gives sentiment polarity of a sentence or an opinion document, not targeted sentiment of the product in the review.

Sentiment analysis is growing in popularity and there is an increased need to perform sentiment analysis. Some popular models used in Opinion Mining are the BiDirectional Encoder Representations from Transformers [5] language model also known as BERT, and Aspect Based Sentiment Analysis also known as ABSA. Currently, many are turning to using BERT or models based on BERT, as it is a great model for a large range of tasks, it is very capable for language inference, and NLP tasks. There are not many easy to use/plug-and play models for Sentiment Analysis. Other people have noticed that there is a lack of a plug-and-play sentiment analysis libraries, and they have created their own collection of sentiment analysis models called pysentimiento [6]. Pysentimiento contains beto a BERT model trained in Spanish, and BERTweet a BERT model trained on English tweets for use with

the English language. Both of these models have been pre-trained for use for evaluating sentiment and emotion analysis. They selected BERTweet as the starting BERT model since it performed a bit better than BERT in their testing. We encountered a similar problem with easy to use models and in their paper they also call out VADER as being one of the few models that you can just install and use.

BERT is frequently used when performing sentiment analysis, so there is a lot of research into how well BERT performs with the default model. There is research to modify the pre-trained BERT model to perform text classification, Chi Sun, Xipeng Qiu, Yige Xu and Xuanjing Huang worked on improving the performance of BERT when performing these text classification tasks [7]. They discovered that some pre-training could significantly boost its performance and that BERT could improve the task with small-size data. Another recent paper has focused on using the pre-trained BERT model and enhancing the BERT model to perform TABSA otherwise known as Targeted Aspect Based Sentiment Analysis [8] which improves on ABSA by focusing on object as sometimes a sentence or document may refer to more than one object. There is active research towards creating smaller, more performant versions of BERT, DistilBERT created by Victor Sanh, Lysandre Debut, Julien Chaumond, Thomas Wolf "a general-purpose pre-trained version of BERT, 40% smaller, 60% faster, that retains 97% of the language understanding capabilities" [9]. They were able to use Knowledge distillation to compress the size of the BERT model, and were able to retain 97% of the capabilities which is pretty incredible. XLnet a generalized autoregressive pre-training method overcomes the limitations of BERT and achieves state-of-the-art performance on sentiment analysis datasets [10].

There is an increase in researching better approaches for ABSA and optimizing how results are calculated, Amit Kushwaha and Shubham Chaudhary use an ML and rule based approach for performing ABSA [11]. Their results had good accuracy

and were able to mitigate some of the noise which is encountered when performing the ABSA extraction which happens when the syntactic rule parses a sentence it was not supposed to. GRACE stands for GRadient hArmonized and CascadEd labeling model, which uses BERT as the backbone [12]. They focused on making sure that GRACE is more resistant to the imbalance of labels, as sometimes a label occurs more often leading to skewed results, GRACE is a great example of building aspect based sentiment analysis models using BERT.

Comparative Opinion Mining

Within *Opinion Mining* is *Comparative Opinion Mining*, a subfield that deals with identifying and extracting information that is expressed in comparative form. Relatively less research has been done in comparative opinion mining. According to Bing Liu and Nitin Jindal, a comparative sentence "expresses an ordering relation between two sets of entities with respect to some common features" [13]. The earliest exploration in this field was conducted by Bing Liu and Nitin Jindal in the same paper. They suggest that comparatives can be classified into two main types: "gradable, and non-gradable" [13]. Non-gradable comparatives are more implicit, while the gradable comparatives are more direct and can be assigned a value like greater than or less than. In COMPA we will be using both types of comparative sentences to form a complete opinion. In Liu and Jindal's research paper they use keywords to filter out sentences that are unlikely to be comparisons, and we take a similar approach to reduce our reviews under evaluation to ensure more relevant data is used in the system. Another challenging aspect of Comparative Opinion Mining is identifying comparative sentences. Nitin Jindal and Bing Liu have looked into the problem of identifying comparative sentences. They started by creating a list of comparative keywords, and checking if these words were present in a sentence [14] which led to a high recall but lower precision. To solve this lower precision

problem, they added Naïve Bayes to the resulting classification to help remove the false positives, and improve precision. In COMPA, we also initially took an approach where we created a list of comparative keywords and then used this list to determine if a sentence is comparative. Instead of adding to this approach, we overcome this problem by performing the analysis using libraries that have overcome this problem such as VADER.

Comparegem is an example of model created to classify the comparative direction and rank entities, it is a generative model based on Gibbs Sampling [15]. In CompareGem, the model classifies each sentence and then is able to generate a conclusion from the summaries. At that time CompareGem outperformed other machine learning approaches such as Support Vector Model (SVM), or Naive Beye's (NB). It does this by modeling two layers of comparative relations, at the sentence level and at the entity pair level [15]. Mirco Franzek, Alexander Panchenko and Chris Biemann created an annotated dataset of comparative sentences and used different approaches for classification, word embeddings, bag of words and concluded that InferSent model performed best [16]. He et al focused on creating a list of comparative words which were used with 6 patterns to classify mobile reviews in Chinese [17]. They were able to achieve high precision and recall but were only focusing on nouns, so they felt that they might've been missing a few things.

COMPA differs from these previous approaches in performing comparative opinion mining, by performing three types of analysis on the reviews to get a comprehensive look to how these two entities compare while also providing evidence for this conclusion. Most of the research has been focused on a singular type of review while COMPA is able to perform its analysis on reviews relating to each entity and reviews directly comparing both entities.

Comparative Systems

There have been a few attempts at creating systems that compare products automatically. Opinion Observer [18] is a system that aims to present a visual comparison of product features. The system primarily focuses on short sentences since Opinion observer focuses on the pro/con review format. This system automatically gathers the reviews from the internet and periodically updates its database. This system uses class sequential rules mining, using a training data set to automatically do feature extraction and comparison between two objects. Most of the data for this system is structured, with the only unstructured data being the free-format reviews. The reviews were also all about the individual products themselves, focusing on displaying difference of scores in common attributes. COMPA instead focuses on the review as a whole instead of just focusing on small sentences, and the review structure can vary as we are gathering reviews from different websites. Another recent attempt in creating a comprehensive comparison search system is Comparative Argumentative Machine otherwise known as CAM. CAM [19] retrieves comparative opinions, and uses a machine learning approach by using InferSent on their data to derive a conclusion if A **better than** B, B **better than** A or A **equal to** B. COMPA in addition to comparative opinions, analyzes reviews only about A, reviews only about B and reviews containing both A and B to determine which product is superior. This approach, which is similar to a human approach, will give a more informed result, when compared to CAM which considers only comparative opinions.

III. METHODOLOGY

COMPA Overview

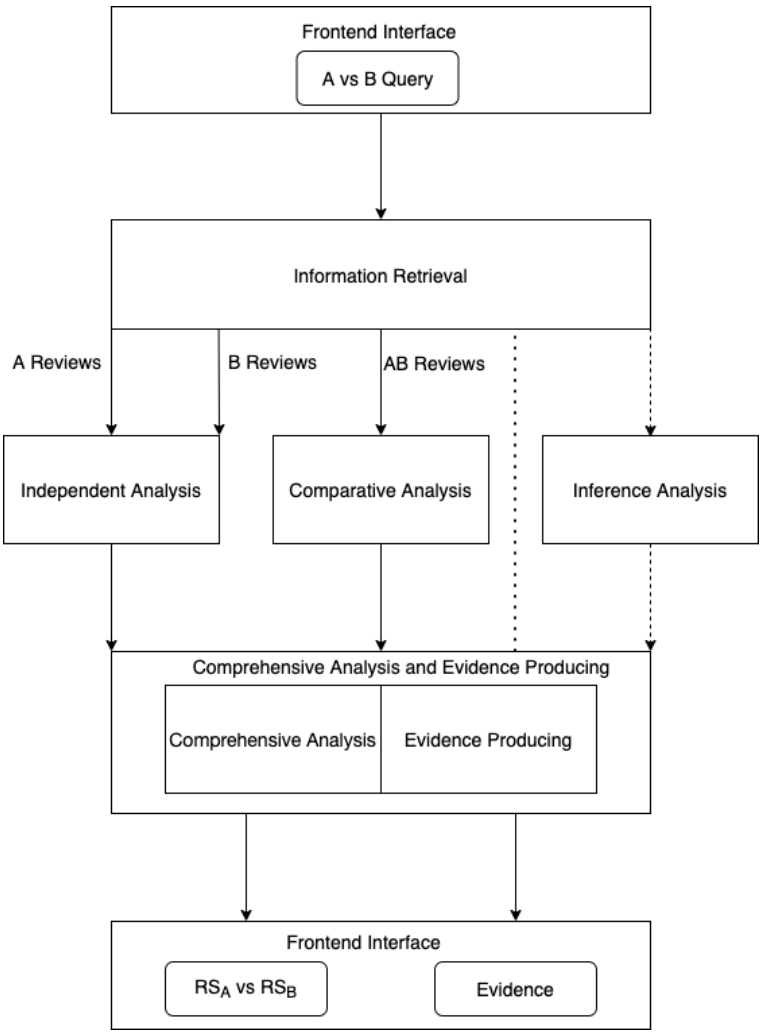

Figure 1: COMPA System Overview

This figure demonstrates the design of COMPA the comparative retrieval and analytical engine. From this figure we can tell the key components, and how the data flows through the system. When designing COMPA, we aimed to perform three types of analysis: Independent Analysis, Comparative Analysis, Inference Analysis. These three analysis types form the three distinct Analysis Components,

Independent Analysis Component, Inference Analysis Component, Comparative Analysis Component which are used later in this system. We start by issuing a query via the Frontend Interface, this is done by issuing a query in the form of A vs B where A and B are distinct products. After submitting the query the user will be presented with a loading icon as the rest of components start to process this query. The Information Retrieval Component takes this query and starts searching the internal dataset to select relevant reviews. The Information Retrieval Component selects reviews related to the query that fit in one of three categories, only A Reviews, only B reviews and only A vs B reviews. The reviews only containing one product, the only A and only B reviews are then sent to the Independent Analysis Component, while the reviews containing A and B are sent to the Comparative Analysis Component. The Independent Analysis Component analyzes the reviews, finding the product's sentiment in each review and categorizing it into either positive, negative or neutral reviews. It then calculates a statistically correct score CS that is used later to compute the final scores. The A vs B reviews are passed to the Comparative Analysis Component which then analyzes the reviews and places them into three categories of A vs B better, B vs A better and A vs B neutral which are used to calculate its own statistically corrected score CS. After these scores are calculated they are then submitted to the Comparative Analysis Component and the Evidence Producing Component to calculate the final scores and generate some evidence supporting the final scores. The Comprehensive Analysis Component computes two final scores RS_A and the resulting RS_B, which represent the percentage of how many people prefer A or B, and the evidence producing engine will produce supporting evidence for this conclusion in a format easy for the user to understand and evaluate. The following sections will go over these components in more detail.

Frontend Interface Component

The Frontend Interface Component is where the user will be able to submit a query and view the final results from COMPA after the reviews have been processed. The user will be able to submit a query to the application in the form of A vs B query where A and B are distinct products, and the source that COMPA should use. They will be able to do this by having two text boxes one for A and one for B, as well as a dropdown that specifies the source YouTube, Amazon, Google as examples and a button to submit the query once the user is done with inputting their selections. After the submit button is pressed the Frontend Interface issues this query to the Information Retrieval Component which will then pass along the retrieved reviews through the rest of the system, which computes a conclusion and returns a result in the format of RS_A, RS_B and evidence. Details about the output returned from COMPA will be defined more later in this paper.

Information Retrieval Component

The Information Retrieval Component is the first part of the analysis components and is the most important part of this system, as it provides the data that flows through the rest of the system. The A vs B query is issued to this Component, and the Information Retrieval Component processes this query by querying its internal dataset with the requested Products A, and B from the selected sources. After it queries the internal dataset, if it doesn't have any data for A or B with the selected sources, the Information Retrieval Component will go and gather new data so that the analysis can be performed. After it retrieves the reviews it needs to ensure that the reviews are relevant to the query, so it has to make sure it only contains A or B and reviews containing both A and B. It filters out reviews that don't contain A, or B or A and B, for example filtering out reviews that

mention the products in the title but don't actually mention the product in the review body. By doing this only relevant reviews should be retrieved from the Information Retrieval Component.

Independent Analysis Component

The Independent Analysis Component is responsible for evaluating the individual reviews. This component receives reviews that only contain A or B, and then analyzes the reviews and place the review result into one of three categories, positive, negative and neutral. It adds up the number of reviews in each category to get values for #positive, #neutral and #negative which are the totals for each category.

Analysis Approaches

To perform the analysis we came up with three distinct approaches. These approaches are used by both the Independent Analysis Component and the Comparative Analysis Component. The three approaches are a simple Natural Language Processing approach, a Machine Learning Approach, and a Natural Language Processing with a sophisticated rule based approach.

The first approach is a simple approach where we perform some Natural Language Processing on the review and use some simple positioning strategies to help with identifying the category for the review. A simple approach for this could be creating a set of adjectives identified as positive, negative and neutral. We could then use these to get an initial idea of the polarity in a sentence, by using a basic system identifying which word came first.

The second approach is to employ some Machine Learning Techniques on the data to perform Comparative Opinion Mining. Some possible approaches that have been used in the field previously have been to use the Naive Beye's Classifier,

Support Vector Model, or a rule-based model. There has been increasing research to use an ML approach for performing Comparative Opinion Mining.

The third approach is similar to the first one, where we use lists of comparative adjectives/adverbs to start classifying the sentence, and then instead of simple positioning strategies we apply a variety of rules to help with this classification. To come up with the rules we would analyze how the adjectives/adverbs were used in sample sentences, and then come up with some rules based on some common patterns that we observed in the sample sentences. We would then be able to use these rules to help with the classification of the sentence.

Score Computation for Independent Analysis Component

For each product A or B, we compute a score S as follows:
We define #total as:

$$\#total = \#positive + \#neutral + \#negative$$

where #positive, #neutral, #negative are the total number of reviews for each category. Therefore, S can be defined as:

$$S = (1 * \#positive + 0.5 * \#neutral + 0 * \#negative)/\#total$$

We then compute a statistically corrected score CS as follows:

$$CS = (S * \#total + 0.5 * c)/(\#total + c)$$

c is a constant representing a threshold for the number of reviews, beyond

which, we would have decent statistical confidence about the score S. c is set based on applications. For our purposes we have set c to 10 as we have a decent number of reviews but not too many reviews.

From the formula we can tell, if $\#total$ is very small, then CS will be close to 0.5 representing neutral. If $\#total = 0$, $CS = 0.5$. If $\#total$ is very big, then CS will be close to S, indicating S is a statistically trustworthy score. If $\#total$ reaches infinity, then $CS = S$.

We compute CS_A for product A and CS_B for product B separately using the 3 numbers for A and the 3 numbers for B.

Then based on CS_A and CS_B, we can compute "Relative Strength based on Independent Analysis" scores $RSIA_A$ and $RSIA_B$ as follows:

$$RSIA_A = CS_A/(CS_A + CS_B)$$

$$RSIA_B = CS_B/(CS_A + CS_B)$$

We can see that $RSIA_A$ and $RSIA_B$ are in the range of 0 to 1, and they add up to be 1.

Comparative Analysis Component

The Comparative Analysis Component focuses on reviews that have a direct comparison between both Product A and Product B. The approach to analyze these reviews is different as we are not computing a simple polarity on the review, but instead focusing on how these two objects are directly compared. Here we employ the same analysis approaches mentioned in the Independent Analysis Component, but apply them to find out how A and B fare when compared to one another.

Score Computation for Comparative Analysis Component

The 3 numbers that our component computes for these AB reviews are as follows:

- $\#A$ (number of reviews where A is better than B)
- $\#T$ (number of reviews where A and B are tied)
- $\#B$ (number of reviews where B is better than A)

Based on the 3 numbers, we can compute "Relative Strength based on Comparative Analysis" scores $RSCA_A$ and $RSCA_B$ as follows:
To compute $RSCA_A$, we use exactly the same formula for CS_A, where

$$\#pos = \#A, \#neu = \#T, and \#neg = \#B$$

To compute $RSCA_B$, we use exactly the same formula for CS_B, where

$$\#pos = \#B, \#neu = \#T, and \#neg = \#A$$

Both $RSCA_A$ and $RSCA_B$ are in the range of 0 to 1, and they add up to be 1. Since the amount of comparative reviews is generally much lower than the amount of reviews we set the value for c much lower than the c value used in Part I, for our purposes we set c to a value of 5.

Inference Analysis Component

The Inference Analysis Component is a key component for when we don't have a direct comparison between A or B, but we can still calculate a comparison between the two by inferring the difference. For example, suppose there is a product A and a Product B that have no direct comparisons, but there is a product C that

is compared with both product A and product B. We can therefore analyze how product A compares to product C and how product B compares to product C. After we get the results of these comparisons we can use them to infer how product A and product B compare. We did not implement this component, as there are many unknowns about what inputs this should take and what this component will output, but this is part of the original design for COMPA, so it is mentioned here.

Comprehensive Analysis Component

After calculating we get these outputs from the other components we need to combine their values to compute a final score to indicate the consumer preference between the two products.

Score Computation for Comprehensive Analysis Component

Now we can combine the 4 scores $RSIA_A$, $RSIA_B$, $RSCA_A$, and $RSCA_B$ to compute two "Relative Strength" scores RS_A and RS_B. We need to use a weight w (in range of 0 to 1, for our purposes this is set to 0.5) indicating how important the comparative analysis is. Then,

$$RS_A = w * RSCA_A + (1 - w)RSIA_A$$

$$RS_B = w * RSCA_B + (1 - w)RSIA_B$$

RS_A and RS_B are our final output scores. They are in the range of 0 to 1, and they add up to be 1. We can now use these numbers as voting from reviewers, they can be interpreted as a percentage from reviewers for example if the values were 0.75 for RS_A, and 0.25 for RS_B, we can say that 0.75 of reviewers would select A over B while only 0.25 of reviewers would select B over A.

Evidence Producing Component

After the calculation of the final two values RS_A, RS_B, the end user also needs to have some evidence for the final result. The Evidence Producing Component focuses on creating evidence that is easy for the user to understand, it selects relevant reviews that were used in the decision-making process, and highlights the words that were key to the comparison result. This evidence should make it easy for the user to understand the final result. For example if the user submitted a query where Product A was 'Coke' and Product B was 'Sprite', and the resulting RS_A value was 0.74, and the resulting RS_B was 0.26, COMPA needs to present some evidence that would show that roughly 74% of reviewers prefer coke. An example of evidence produced by the Evidence Producing component would be as follows: "Coke tastes **better** than Pepsi, and is often **cheaper** and more **available** in more places." This would be one of many supporting evidences for this conclusion, which would inspire more confidence in COMPA's result. This evidence with keywords in the comparison being underlined, and easily visible allows the user to gain much more confidence in the final result. Since the result is a ratio, some evidence from this component will favor the other product. For example there may be a sentence like "Pepsi has more **variety** than Coke", which would help inspire confidence with the user as Product A is preferred more than Product B, but some still prefer Product B, so having this evidence helps inspire more confidence as the evidence is not lopsided towards one direction but rather demonstrates that the result is correct. The Evidence Producing Component produces these sentences from the original reviews that were used in calculating the scores in the other components.

IV. IMPLEMENTATION

In the following sections we cover how COMPA is implemented.

Dataset Generation and Processing

When first approaching this problem there was barely any data available except for an Amazon reviews dataset published in the Stanford SNAP website [20]. We loaded this initial dataset but found that the reviews only went up to the year 2013, which was difficult to use as we would be limited to products released up to that year. With the problem of not having a suitable dataset at the time of the project's inception we discussed having three distinct sources, Articles from Google Search, YouTube Video Captions, and Amazon Reviews. We designed COMPA to be able to gather results from each of these sources as requested, and scrape the web to gather more reviews if the internal dataset has no available data for the given query.

Scraping the Web

As part of building COMPA the capability to gather data when requested was critical, so we created a service that is able to scrape the web as well as download articles from the web. It leverages two libraries Python 3 libraries Newspaper3k, and BeautifulSoup. Newspaper3k is used to remove all the HTML tags from a web page and only return the written text on the page, it works decently well at removing most tags although sometimes the data it gets back has some leftover tags. BeautifulSoup is used when scraping Amazon reviews to help with extracting only Amazon reviews, by selecting specific HTML tags. This system finds articles on the web by performing a Google search on the two items, it then looks for YouTube reviews where both A and B are mentioned in the title. We are then able to look at

the caption data for that video and use that as a potential review. For Amazon we are able to search and gather Amazon reviews from Product Pages that contain A or B. In order to perform this web scrapping Python 3 uses Selenium to invoke a version of headless chrome that helps with bot detection while also helping with rendering the page on a browser for better results.

Frontend Interface Component Implementation

The Frontend Interface is implemented as it was described in the methodology chapter, with two text boxes labeled for Product A, Product B and a dropdown to select the Source(s) for the comparison. The result is displayed in two boxes one that contains all the final score values, how many reviews were used and which sources were used for this conclusion with another box being used to display the resulting evidence. The Frontend Interface is built using Angular a web framework developed by Google to create responsive Single Page Applications. Angular uses Typescript and to have a responsive web design with minimal styling, the Angular-Material library is used. Angular Material follows the Material UI design principles, and is very similar to other CSS libraries like Bootstrap that provide some reusable components. Angular is used as it is easy to create a web application, and to make HTTP service calls and render that on the page easily.

Information Retrieval Component Implementation

The Information Retrieval Component is implemented very similarly to how it was described in the methodology chapter. A query for two products is submitted in the format of A vs B where A and B are two entities (e.g., consumer products models) that are going to be compared to the system, along with the selected data sources. The Information Retrieval Component searches for reviews in its internal dataset that are from the selected data sources, Amazon, Google and YouTube and

contain A or B or both A and B. It is designed to gather data in real time but to improve performance it caches the latest results for each source and retrieves them based on A or B, and performs some data refreshing asynchronously. After it has retrieved the reviews it categorizes all the reviews into three categories, reviews containing only A, reviews containing only B and reviews containing both A and B. In Methodology it is mentioned that this system filters out reviews that do not contain A or B in the review body, and that feature is implemented in this Information Retrieval Component. An example of filtering out reviews is when using cell phones as product A and B. Cell phones have many reviews available, but as they are cell phones they also have many accessories which means that we get a lot of accessories(e.g., phone cases) instead of the phone. Currently, to avoid this a simple method is implemented to check for products where the product name doesn't contain one of the phrases in a list of small stop words that are commonly used when describing accessories, and also checking that the product is mentioned in the review. The stop list was created after observing that many of these products for example cases would commonly use the same words over and over, these words being 'and', 'for', and 'with'. This approach helps with filtering out some reviews that don't talk about the product which helps with keeping the reviews relevant to the query, and even though they are a small list of words they have filtered out many reviews. This approach is simple and works well for most reviews but in some occasions if a product's name is used in a different context you might get some not relevant review results, this will be discussed more in depth in the experiments section.

The Information Retrieval Component leverages Java 8 and the Spring Boot framework to provide a REST API for gathering relevant information to perform Comparative Opinion Mining. It communicates directly with a MySQL database and an Elasticsearch database. Our Elasticsearch database is used to extract

relevant reviews, so Amazon reviews, YouTube captions and internet articles are stored in Elasticsearch. The MySQL database is used to keep track of some metrics like what has been searched and how often, and it is often used as a persistent cache so that new versions of COMPA do not need to rebuild their cache layer. The Information Retrieval Component also talks to a microservice written in Python 3 which is able to scrape Amazon reviews and retrieve different articles from Google Search Results. After it retrieves the reviews it returns them in a JSON format to be processed by the next components.

Independent and Comparative Analysis Component Implementation

As mentioned in the methodology section the Independent Analysis Component and the Comparative Analysis Component both leverage the same approach but have different inputs. As mentioned in the methodology, there are two possible approaches to calculate the initial results that would be used in score calculations. Initially we worked on implementing a simple NLP approach with rules. This initial approach worked fairly well, and was extremely simple to implement but quickly found that this process itself would require more research to improve the implementation. An example of more research is that we would need a good set of comparative adjectives and grammar rules for different situations, which would take a while to implement/research. Instead, we focused on implementing our ML approach with some easy to use libraries to perform the actual ML analysis. The two libraries we looked into using were VADER and Aspect Based Sentiment Analysis or ABSA for short. Both Vader and ABSA were used to perform analysis on the same dataset as we were unsure about which would perform better, ultimately we found that ABSA is a better library for our purposes. As mentioned in the methodology section, there are a lot of unknowns for the Inference Component and for our purposes we did not implement the Inference Component.

In each component we will still describe the approaches for both VADER and ABSA to provide a clearer picture for why we used ABSA. Both Vader and ABSA are available as Python Libraries so the Independent Analysis Component and the Comparative Analysis Component are both implemented in Python 3 using the FLASK framework to provide a REST API.

VADER

Valence Aware Dictionary for sEntiment Reasoning or VADER [21] is a simple rule-based model based on a valence-based, human-curated gold standard sentiment lexicon. VADER is fast and works well on social media style text and readily generalizes to multiple domains, so we are able to use it to analyze our reviews. It was built and tested against many state of practice benchmarks and even outperforms humans at rating sentences with an overall F1 classification accuracy of 0.96 versus a human's accuracy of 0.84 on the same data [21]. VADER improves on Linguistic Inquiry and Word Count (LIWC) dictionary which was a previous gold standard, by focusing on sentiment analysis and ensuring that it has high accuracy. VADER focuses on the polarity of the sentence and the intensity of the emotion in the sentence.

ABSA

The other library we are using is using targeted sentiment analysis, also known as Aspect Based Sentiment Analysis (ABSA), which aims at detecting fine-grained sentiment polarity towards targets in a given opinion document [22]. The ABSA open source library has a ready to use model that consists of the BERT language model, which provides features and the linear classifier [23]. BERT is chosen for how it performs it's next-sentence prediction, which is done as a sequence-pair classification. BERT is very successful in various tasks due to "massive pre-training

on large amounts of unlabeled text" [24], meaning that it is an excellent choice to be used as part of the ABSA library. One of the key issues with using BERT is that the model can approximate and find patterns that don't reflect how language is actually used. The author of the library published an article [23], where he documents the flaws with using BERT by itself for sentiment analysis, and notes that we could train with better data but that it could still result in the model being skewed. The author proposes using a new tool that can be used while training called the Professor which helps tweak the model in real time to help prevent this model from skewing. We are able to leverage a model that the author has published which is the default because it has better performance, and is trained from the restaurant's dataset from the Sentiment Analysis Workshop, Semeval'14. ABSA has a few issues while using it for Sentiment Analysis, one is that the default model works out of the box, but the author of the Absa library says it is preferred to re-train the model with your data. Another issue is that the pre-trained model has some issues with some reviews, so we are losing some data which will be evident in the experiments section.

Independent Analysis Implementation

Our Independent Analysis Component takes reviews containing only A or B as inputs and then passes the value into the two libraries. Our approach when using **VADER** is that we go through all the reviews containing only A or B, and pass in the review into VADER and get the sentiment and place it in one of the categories of positive, negative or neutral. The other approach when using **ABSA** is to pass the review as the text being analyzed and using the Product as the aspect. Then we get the sentiment value and add it to a list of positive, negative or neutral.

After the libraries are done processing we are able to take the resulting values and compute the statistically corrected score CS and the 'Relative Strength based on Independent Analysis" scores $RSIA_A$ and $RSIA_B$ which can then be passed into

the Comprehensive Analysis Component for final calculations.

Comparative Analysis Component Implementation

Our Comparative Analysis Component takes in reviews directly comparing A and B as inputs and then passes these reviews into the two libraries for analysis.

For **VADER** the approach is a bit complex, as VADER is designed to calculate the sentiment for a sentence, so we had to employ our own changes to use it for performing direct comparison analysis. We go through and gather all the sentences that directly compare Product A and Product B, and then pass each sentence into VADER to calculate the sentiment. We then use the positioning of the two products in the sentence to calculate how they compare. For example given a sentence 'Nintendo Gameboy has longer lasting battery life than the Sega Game Gear', VADER would say this is a positive sentiment sentence, and if we were comparing 'Nintendo Gameboy' as Product A and 'Sega Game Gear' as Product B then we would say that Product A is better than Product B, as Product A is first in the sentence and add it to that list. If Product B was in front then we would add it to the list for Product B is better than Product A. Otherwise if it is a neutral sentiment we would add it to the Neutral A vs B list. Then depending on which list had more sentences we award that review to that category, and add that list of sentences to the final list. While we are traversing through the reviews sentence by sentence to gather sentences about Product A and Product B, we can create groupings of sentences that were only about Product A and only about Product B to enhance the numbers for the individual analysis. We can do this by reading through the full review sentence by sentence and if the sentence mentions Product A then we assume all sentences afterwards are about Product A until we encounter a sentence that talks about Product B or a sentence that talks about A and B. We do the same for sentences for Product B. We gather all the sentences that were

talking about just Product A or just Product B and pass that combined sentence into Vader to compute polarity. After these values are computed we can add the Product A or Product B review values (positive review, negative review, neutral review) from the A vs B reviews and add them to A or B in their respective slots.

In comparison, the process for using **ABSA** is significantly simpler when directly comparing A and B reviews. To use ABSA we pass in the full review, and use A and B as the two aspects we are comparing against. We then compare the output for each aspect. If the review is Positive for A and A is first then we add it to a list of positive A vs B, If the review is Positive for B and B is first then we add it the list for B is better than A, Otherwise if both aspects are neutral then it is a neutral review. After we perform this analysis with both libraries, we now have an output of 3 numbers that we're able to use to calculate "Relative Strength based on Comparative Analysis" scores $RSCA_A$ and $RSCA_B$ as mentioned in the methodology section and using the formula specified there. These values can then be used by the Comprehensive Analysis Component to compute the final result.

Comprehensive Analysis Component

The Comprehensive Analysis Component is implemented using Python 3, and as mentioned in methodology is in charge of taking the scores computed in the other components to create a final result. This component performs the score calculation, and rounds the numbers to two decimal places to make it easier for the user to visualize the final result. It adds the values to an array that is added to the final JSON returned to the Frontend Interface to be displayed elegantly.

Evidence Producing Component Implementation

In our implementation the Evidence Producing Component is very basic due to how the libraries we used to perform the different kinds of analysis work, where they

output a sentiment but no evidence for how that sentiment is calculated. We still implemented an Evidence Producing Component, so there would be some evidence to help support the conclusion. This component starts by gathering the direct comparisons that directly support the conclusion and then adds supplementary information from the individual reviews. For example if we had two products Product A as 'Sprite' and Product B as 'Coke', and the review conclusion was an RS_A value of 0.65 meaning that 65% prefer 'Sprite' then this Component would first gather reviews where Sprite is better than Coke in the same review, and then grab reviews where the Sprite is talked about positively. This component gathers 20 results for each query, and limits them to reviews that only have 6 or fewer sentences so that the user is able to read all the information and not be overwhelmed. Longer reviews could be used for evidence if we had the capability to highlight keywords so that it would make the reviews easier to read. Counting sentences is done by leveraging the Python library called the natural language toolkit or nltk for short, which is used for natural language processing. The user is still presented with some reviews and the UI is set up to have a button to gather more interviews, so in the future a user could click on the button to load more reviews, currently this button does not do anything. These reviews are presented in the Front-end Interface Component in their own box, with some spacing between each review so that the user can review the outcome.

V. DEMONSTRATION AND EXPERIMENTATION

Demonstration

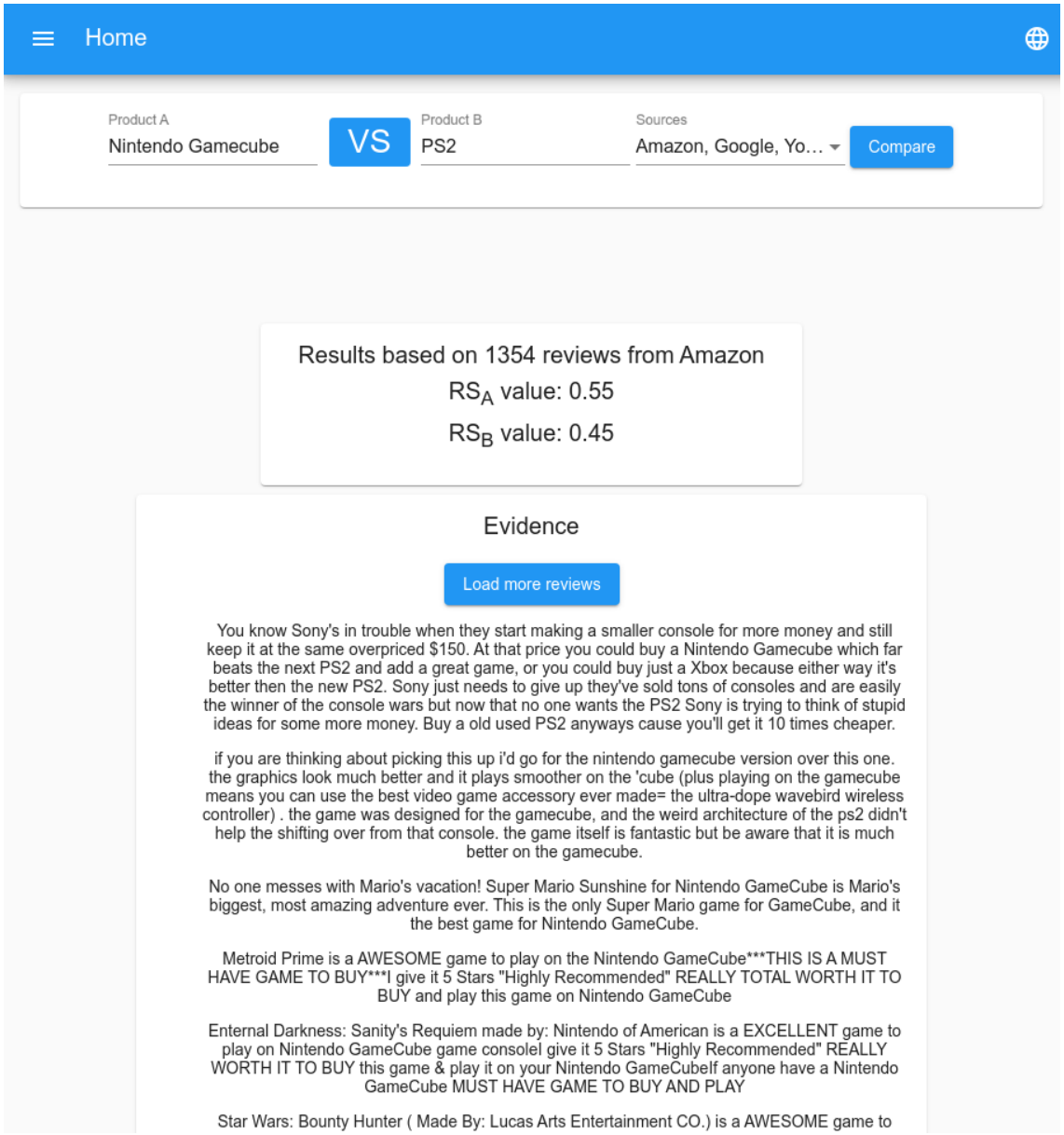

Figure 2: COMPA UI with Evidence

This picture demonstrates the Front end Interface for COMPA after a query is issued and results are computed. In this given query Product A is 'Nintendo

Gamecube' and Product B is 'PS2'. From the picture we can tell that the sources requested were Amazon, Google and YouTube, but we only got results from Amazon, and we know that there was a total of 1354 reviews from Amazon used in these calculations. We are also able to see some Evidence for the result, in its separate box. Within the evidence box each paragraph separated by white space is another different review. We can tell that the first two reviews are evidence from the direct comparison reviews, and the remaining reviews are all from the individual reviews. From reading the reviews, most of them are talking about the 'Nintendo Gamecube' positively, which reflects the outcome of this comparison as calculated by COMPA. The first two reviews are the most crucial reviews as they are the direct comparisons and talk about the Product A and B directly, so a user is able to get more specific info on how the two products compare. The other reviews while talking positively about the 'Nintendo Gamecube' talk more about the games that play on the product instead of the product itself. In this specific case this works fairly well as the 'Nintendo Gamecube' is a video game console, so talking about the games that are playable on the console works great because it is the key functionality of this product. This result also demonstrates how the two values for RS_A, RS_B add up to 1, and is a good example of the values being close but not too close together. This demonstrates how easy it is to issue a query to COMPA, and how the evidence helps with understanding the final scores. With this evidence we can tell that some reviews are talking about the 'Nintendo Gamecube' while others are talking about 'PS2', which matches with our end result with roughly 55% preferring the 'Nintendo Gamecube' while 45% preferred the 'PS2'.

Experiments

These experiments were run on a Desktop PC with an AMD Ryzen 5 3600 processor a 6 core CPU with 12 threads, 16GB of DDR4 3200 RAM and using Pop

OS 21.04, a Linux Distribution based on Ubuntu. COMPA and all of its components were running in a Docker container which was able to fully utilize all the system resources, both databases were also present in this container. The Python libraries were only able to use the CPU, as the GPU would frequently run out of memory as it was a GPU designed for Gaming and not performing Machine Learning tasks.

Comparing VADER and ABSA

To choose a library to use for COMPA we needed to evaluate ABSA and VADER and see how they performed on our data. To do this we selected five different queries to test how ABSA and VADER performed with different queries. We tried a few specific queries using the product names, some queries using part of the product name and some broad queries to see what the Information Retrieval Component would return and how ABSA and VADER would process these reviews. To make this analysis consistent we use the same five queries in the following analysis sections.

Table 1: Total Reviews Processed by ABSA and VADER

Review Query	# of Reviews by VADER	# of Reviews by ABSA	Total number of reviews available
vive vs rift	1,040	832	1,092
xbox one vs ps4	2061	832	3006
star trek vs star wars	2,124	1,574	2,160
pepsi vs coke	572	306	578
ps3 vs xbox 360	2663	2,661	3,002

First we started by issuing queries to COMPA, seeing how many reviews the Information Retrieval Component retrieved and then evaluating how many reviews ABSA and VADER processed. The table reveals that VADER is able to find a sentiment for most of the supplied reviews including finding neutral or no sentiment, but ABSA is only able to analyze some of the reviews. This is likely due to using

the default model, so we are losing some possible analysis since we have not trained ABSA using our data. In the next table we can see precision and recall for ABSA and VADER using the same sample queries. In our experiments through finding good sample queries we discovered that being more specific returned less reviews and going more broad led to retrieving more reviews. In general the amount of comparative reviews was usually less than the reviews retrieved for the individual products.

Precision + Recall

After looking at how many reviews were processed, we now shift our focus on to how well the ABSA and VADER performed, as being able to process a lot of reviews isn't good if we don't have a good precision or recall value. To make processing the total amount of information easier we opted to select the same amount of reviews for five sample queries, so we are selecting 100 reviews for each category (only A, only B and only A and B), leading us to having 300 reviews total. This next table is similar to the previous table and will show us how many were processed by both algorithms.

Table 2: Total Reviews Processed by ABSA and VADER out of 300 Reviews

Review Query	# of Reviews by VADER	# of Reviews by ABSA	Total number of reviews available
vive vs rift	273	222	300
xbox one vs ps4	261	267	300
star trek vs star wars	278	244	300
pepsi vs coke	282	235	300
ps3 vs xbox 360	271	261	300

Upon this more consistent data set, we are able to see that VADER still classifies more but ABSA is closer than the previous table. Now that we have a more consistent data set we are able to focus on the correct classification of

sentiment in a review. This next table shows the results,in this case we view a correct classification of sentiment as positive and the correct classification of a review as neutral is viewed as a negative. In order to compute these classifications I had to pull up the individual sentences as grouped by the algorithms and had to score these by hand, which was a very time consuming process.

Table 3: VADER and ABSA Precision + Recall for Overall Sentiment Classification

Library	**VADER**		**ABSA**	
Review Query	`Precision`	`Recall`	`Precision`	`Recall`
vive vs rift	0.85	0.95	0.94	0.84
xbox one vs ps4	0.86	0.98	0.90	0.78
star trek vs star wars	0.65	0.99	0.92	0.92
pepsi vs coke	0.73	0.99	0.86	0.43
ps3 vs xbox 360	0.81	0.97	0.82	0.88

From this table we can immediately see that although VADER has processed more reviews, that ABSA has higher precision in it's classification of sentiment in comparative reviews. As far as why VADER achieved such high recall in the Star Trek and Star Wars was due to there being a small amount of neutral reviews. This is corrected in the next table.

This next table focuses on the correct classification of sentiment, so for the purposes of calculating these values we focused on just the positive and negative sentiment, and we now ignore the correct classification of neutral reviews.

Table 4: VADER and ABSA Precision + Recall for Correct Sentiment Classification

Library	**VADER**		**ABSA**	
Review Query	`Precision`	`Recall`	`Precision`	`Recall`
vive vs rift	0.90	0.92	0.77	0.82
xbox one vs ps4	0.86	0.97	0.94	0.94
star trek vs star wars	0.82	0.73	0.89	0.98
pepsi vs coke	0.77	0.88	0.93	0.91
ps3 vs xbox 360	0.80	0.94	0.85	0.94

With this comparison we are able to see that ABSA has higher precision than

VADER but at a cost as ABSA processing takes longer to process the queries, meaning most of the time is spent waiting on the ABSA library to run on all the data. ABSA also has higher recall so it is able to gather more relevant documents, and correctly classify them.

One notable thing when evaluating the two libraries and their performance was that on when using the more broad queries, there was definitely a distinct drop of relevant individual reviews. For example there were a few reviews that had nothing to do with Pepsi or Coke, as there were some books where those were the character's names or books referencing the company instead of the product. This led to an inaccurate conclusion for pepsi vs coke, but helped with evaluating the performance of ABSA and VADER effectively. A similar thing happened with vive and rift, if you use the more specific query you get more relevant data but if you switch to a more broad query you start getting those words as part of speech instead of the products that were requested. As a result of these experiments, we discovered that ABSA was a better library to use for COMPA despite the amount of time it took for the queries to run.

COMPA Verification

After testing the individual libraries, we needed to test COMPA's output and validate that it matched our expectations. For these experiments we ran a series of queries and evaluated if their results were good results based on the evidence displayed. We also took a note on how long these took to process these queries. For matches expectations we have 'Y' for Yes, 'N' for No, and 'M' for Mostly where it's mostly true.

This table demonstrates the various level of performance with COMPA's performance. Here we had 5 queries that we agreed with the conclusion, 3 that we felt were mostly correct and 2 that definitely were not correct based on the

Table 5: COMPA Output Verification: Queries, results and processing time

Review Query	Matches Expectations	Total # of Reviews	Total Processing Time
HTC Vive vs Oculus Rift	Y	1093	13 mins 15 seconds
Xbox One vs PS4	Y	3006	33 mins 50 seconds
Star Trek vs Star Wars	Y	2343	27 mins 36 seconds
Pepsi vs Coke	N	1307	12 mins 11 seconds
PS3 vs Xbox 360	Y	3002	33 mins 53 seconds
iPad Mini vs Kindle Fire	N	2070	23 mins 38 seconds
Sony WH1000XM4 vs Airpods Max	M	97	2 mins 43 seconds
iPhone 12 vs iPhone 13	M	357	9 mins 54 seconds
iPhone X vs iPhone XS	N	364	5 mins 25 seconds
Nintendo Gamecube vs PS2	Y	1354	16 mins 30 seconds

evidence. These were all evaluated by me by running the query, looking at the resulting values for RS_A, RS_B and reading the evidence and seeing if the evidence matched the final scores RS_A, RS_B.

Just like finding written reviews for the products that you are searching for, COMPA downloads a lot of reviews for the given queries but not every review is stellar. A common issue is that just because a review mentions a product doesn't mean that the review is about that product. So after reading the evidence you can get a feel for how accurate COMPA's response is after reading the first five reviews. Overall there is reasonable confidence that COMPA's calculation is correct, especially if you use a more specific name instead of a more common/broader name. Another thing to note is that if you go much broader than COMPA will process a lot larger sample of reviews, so we have a higher probability of getting a more correct answer, although we also have a higher probability of getting a worse answer. Most of the ones that we agreed with are the ones that had a lot larger

amount of reviews, with an exception being Pepsi and Coke. Pepsi and Coke we disagreed with mainly because there was a lot of values for a book where Pepsi and Coke are the primary characters, which lead to a lot of the reviews being skewed/miscalculated. In the future the Information Retrieval Component needs to do a more complex job of filtering our reviews that are really not relevant to the query, which is in its own an interesting research problem, with this simple solution working well most of the time but improving could lead to improved results.

Another aspect to look at is the processing time for these reviews, which from the table we can see increases greatly the more reviews it needs to process. From my experience running the queries and looking at how COMPA was processing the queries, the version of BERT could probably perform better and process more reviews, currently ABSA is trained on smaller reviews, so it is not processing all the reviews that the information retrieval component retrieves we are losing some reviews due to this limitation. Overall this library processes a decent amount of reviews and is processing them relatively well. Although the processing time is initially long, after we complete the initial processing we are able to cache the result so that the subsequent results are returned in a few seconds, so that later queries are able to return much quicker.

VI. CONCLUSION

In this thesis work, we proposed and implemented COMPA, a novel comparative retrieval and analytical engine in the context of consumer products. Potentially the system can have great practical significance as consumers typically spend lots of time reading reviews and comparing various models before an online purchase. COMPA is designed to perform comprehensive analysis integrating three components of independent, comparative, and inference analysis. We evaluated the performance of COMPA on real datasets, demonstrating its utility and promise. Comparison is ubiquitous in real life. Although COMPA is implemented for consumer products, the system and methodology can be easily adapted to other application domains involving online comparative decision-making.

There are many directions for future work beyond our preliminary effort in COMPA. For example, we designed but didn't implement inference analysis, which can improve analysis accuracy by leveraging indirect comparisons. In general, how to optimize accuracy is a major research focus and many machine learning, data mining and natural language processing techniques can be considered. Another important research focus is to optimize query processing time because the user will be waiting online for analysis result after issuing a comparison query. One possible solution is to provide the user with alternative analysis accuracy and processing time trade-offs. Currently COMPA is designed to perform a comparison between two products. Such pairwise comparisons can be extended to perform evaluation for a set of products, for example, by aggregating partial rankings into a total ranking.

REFERENCES

[1] B. Liu, "Sentiment analysis and opinion mining," *Synthesis Lectures on Human Language Technologies*, vol. 5, no. 1, pp. 1–167, 2012.

[2] K. D. Varathan, A. Giachanou, and F. Crestani, "Comparative opinion mining: A review," *Journal of the Association for Information Science and Technology*, vol. 68, pp. 811–829, Apr 2017.

[3] L. Zhang, S. Wang, and B. Liu, "Deep learning for sentiment analysis : A survey," *CoRR*, vol. abs/1801.07883, 2018.

[4] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey.," 2020.

[5] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018.

[6] J. M. Pérez, J. C. Giudici, and F. M. Luque, "pysentimiento: A python toolkit for sentiment analysis and socialnlp tasks," *CoRR*, vol. abs/2106.09462, 2021.

[7] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune BERT for text classification?," *CoRR*, vol. abs/1905.05583, 2019.

[8] C. Sun, L. Huang, and X. Qiu, "Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence," *CoRR*, vol. abs/1903.09588, 2019.

[9] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter," *CoRR*, vol. abs/1910.01108, 2019.

[10] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *CoRR*, vol. abs/1906.08237, 2019.

[11] A. Kushwaha and S. Chaudhary, "Review highlights: Opinion mining on reviews: A hybrid model for rule selection in aspect extraction," in *Proceedings of the 1st International Conference on Internet of Things and Machine Learning*, IML '17, (New York, NY, USA), Association for Computing Machinery, 2017.

[12] H. Luo, L. Ji, T. Li, N. Duan, and D. Jiang, "GRACE: gradient harmonized and cascaded labeling for aspect-based sentiment analysis," *CoRR*, vol. abs/2009.10557, 2020.

[13] N. Jindal and B. Liu, "Mining comparative sentences and relations.," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 2, (Department of Computer Science, University of Illinois at Chicago), pp. 1331–1336, 2006.

[14] N. Jindal and B. Liu, "Identifying comparative sentences in text documents," in *Proceedings of SIGIR-06, the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 244–251, ACM Press, January 2006.

[15] M. Tkachenko and H. W. Lauw, "Generative modeling of entity comparisons in text," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, (New York, NY, USA), p. 859–868, Association for Computing Machinery, 2014.

[16] M. Franzek, A. Panchenko, and C. Biemann, "Categorization of comparative sentences for argument mining," *CoRR*, vol. abs/1809.06152, 2018.

[17] S. He, F. Yuan, and Y. Wang, "Extracting the comparative relations for mobile reviews," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pp. 3247–3250, 2012.

[18] B. Liu, M. Hu, and J. Cheng, "Opinion observer: Analyzing and comparing opinions on the web," in *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, (New York, NY, USA), p. 342–351, Association for Computing Machinery, 2005.

[19] M. Schildwächter, A. Bondarenko, J. Zenker, M. Hagen, C. Biemann, and A. Panchenko, "Answering comparative questions: Better than ten-blue-links?," in *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval*, CHIIR '19, (New York, NY, USA), p. 361–365, Association for Computing Machinery, 2019.

[20] J. Leskovec and A. Krevl, "SNAP Datasets: Stanford web data: Amazon reviews." `https://snap.stanford.edu/data/web-Amazon.html`, June 2014.

[21] C. J. Hutto, C. J. Hutto, and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text vader: A parsimonious rule-based model for sentiment analysis of social media text," -01 2015.

[22] J. Pei, A. Sun, and C. Li, "Targeted sentiment analysis: A data-driven categorization," *ArXiv*, vol. abs/1905.03423, 2019.

[23] R. Rolczyński, "Do you trust in aspect-based sentiment analysis? testing and explaining model behaviors," 2021.

[24] J. Kamps, N. Kondylidis, and D. Rau, "Impact of tokenization, pretraining task, and transformer depth on text ranking," in *TREC*, 2020.