HAW
HAMBURG

BACHELOR THESIS
Nhat Khanh Huy Tran

Development of a Machine Learning System for Aspect-Based Sentiment Analysis and Text Summarization of Video Game Reviews on Steam

FACULTY OF COMPUTER SCIENCE AND ENGINEERING
Department Computer Science

Fakultät Technik und Informatik
Department Informatik

HAMBURG UNIVERSITY
OF APPLIED SCIENCES

Hochschule für Angewandte
Wissenschaften Hamburg

Nhat Khanh Huy Tran

Development of a Machine Learning System for Aspect-Based Sentiment Analysis and Text Summarization of Video Game Reviews on Steam

Bachelor Thesis based on the examination and study regulations
for the degree programme
Bachelor of Science Angewandte Informatik
at the Department of Computer Science
of the Faculty of Engineering and Computer Science
of the University of Applied Sciences Hamburg

Supervising examiner: Prof. Dr. Marina Tropmann-Frick
Second examiner: Prof. Dr. Stefan Sarstedt

Day of delivery: 31. August 2021

Nhat Khanh Huy Tran

Title of Thesis

Development of a Machine Learning System for Aspect-Based Sentiment Analysis and Text Summarization of Video Game Reviews on Steam

Keywords

Aspect Extraction, Sentiment Analysis, Aspect-Based Sentiment Analysis, Text Summarization, Data Visualization, Video Games, User Reviews, Unsupervised Learning

Abstract

As video games' popularity as a form of entertainment grows, so as the difficulties in dealing with the enormous amount of reviews generated by users. NLP (Natural language processing) techniques can be used to summarize the opinions in these reviews, which helps developers immensely in understanding customers and supports potential customers' buying decisions. This study introduces SteamInsider - an unsupervised machine learning system for aspect-based sentiment analysis, text summarization, and data visualization of video game reviews on Steam[a]. The system uses a simple rule-based approach to extract the most important keywords from a game's user reviews and cluster them into groups, which are called aspects. We then perform sentiment analysis on each sentence that belongs to an aspect using a pre-trained RNN-based sentiment classifier. Based on reader's preferences like aspects, polarities, time, etc., certain sentences are extracted, and then clustering methods are used to identify salient opinions and generate a customizable extractive summarization. A dashboard for visualizing the results and further complex analyses is also created. The unsupervised system shows great usefulness and efficiency, which can potentially be applied to other domains like mobiles app, product, hotel or restaurant reviews, etc.

[a] https://store.steampowered.com/

Nhat Khanh Huy Tran

Thema der Arbeit

Entwicklung eines Machine-Learning-Systems zur aspektbasierten Sentiment-Analyse und Textzusammenfassung von Videospiel-Rezensionen auf Steam

Stichworte

Aspektextraktion, Sentimentanalyse, Aspektbasierte Sentimentanalyse, Textzusammenfassung, Datenvisualisierung, Videospiele, Benutzerrezensionen, Unüberwachtes Lernen

Kurzzusammenfassung

Mit der Popularität von Videospielen als Unterhaltungsform wächst auch die Schwierigkeit, mit der enormen Menge an Bewertungen umzugehen, die von den Nutzern generiert werden. NLP-Techniken (Natural Language Processing) können verwendet werden, um die Meinungen in diesen Bewertungen zusammenzufassen, was Entwicklern immens hilft, Kunden zu verstehen und Kaufentscheidungen potenzieller Kunden zu unterstützen. Diese Studie stellt SteamInsider vor – ein unbeaufsichtigtes maschinelles Lernsystem für aspektbasierte Stimmungsanalyse, Textzusammenfassung und Datenvisualisierung von Videospielbewertungen auf Steam[a]. Das System verwendet einen einfachen regelbasierten Ansatz, um die wichtigsten Schlüsselwörter aus den Benutzerbewertungen eines Spiels zu extrahieren und sie in Gruppen, die als Aspekte bezeichnet werden, zu gruppieren. Wir führen dann eine Sentiment-Analyse für jeden Satz durch, der zu einem Aspekt gehört, indem wir einen vortrainierten RNN-basierten Sentiment-Klassifikator verwenden. Basierend auf den Vorlieben des Lesers wie Aspekten, Polaritäten, Zeit usw. werden bestimmte Sätze extrahiert und dann werden Clustering-Methoden verwendet, um auffallende Meinungen zu identifizieren und eine anpassbare extraktive Zusammenfassung zu generieren. Außerdem wird ein Dashboard zur Visualisierung der Ergebnisse und weiteren komplexen Analysen erstellt. Das unbeaufsichtigte System zeigt eine große Nützlichkeit und Effizienz, die möglicherweise auf andere Domänen wie mobile Apps, Produkt-, Hotel- oder Restaurantbewertungen usw. angewendet werden kann.

[a] https://store.steampowered.com/

Contents

List of Figures

1 Introduction

1.1 Opinion summarization of video games reviews

Gaming is currently the world's favorite form of entertainment, as the gaming industry generated more revenue than TV, movies, and music did last year. The gaming sector is estimated to be worth $159.3 billion in 2020, which is a sizeable increase of 8.5% from 2019. Current forecasts estimate the video games industry to be worth $268.8 billion by 2025. [8]

As the popularity of video games grows, so as the competition between video game companies. Publishers and developers often struggle to find out what gamers really want and how to best cater to their needs and interests. Thus, it is important for them to understand the strengths and weaknesses of not only their own products but also of the most successful titles. Additionally, the opinions of other people about a game can significantly affect potential customers' buying decisions. Video games reviews offer user-generated data that can be processed and studied in order to identify both people's concerns and the user-perceived quality of the game. [25]

A number of video game retailers, distributors (Steam, GOG[1], etc.) offer a wide selection of games, spanning various genres. By visiting such stores, not only are people able to look through a game's description and its features, delve into the reviews of the game provided by other users and experts, but they can also contribute their own reviews. As some of the most popular titles can have millions of reviews, manually going through all of them and extracting the relevant information is an insurmountable task. Thus, the large scale of information poses the need and challenge of building a system that can automatically summarize users' opinions in those reviews.

[1] https://www.gog.com/

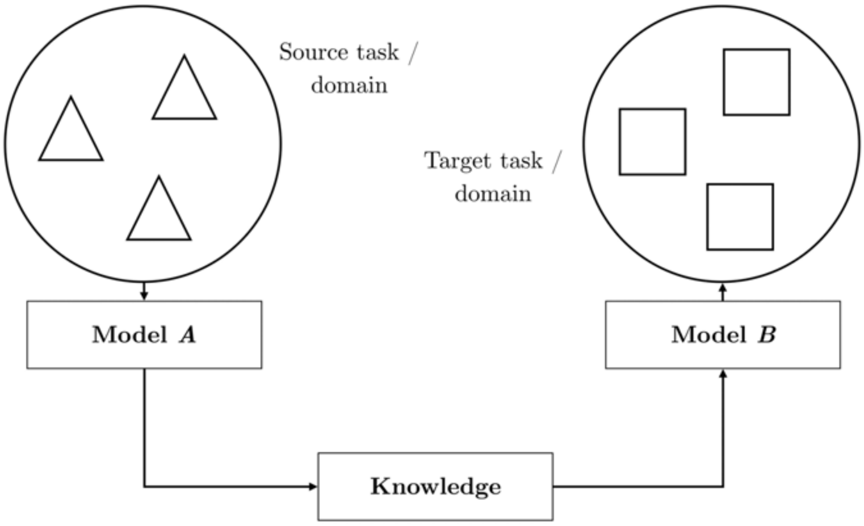


Figure 1.1: The process of transfer learning [29].

1.2 Transfer learning in natural language processing

Acquiring enough high-quality labeled data is crucial for training good machine learning models, but it is costly and time-consuming, especially in the realm of NLP, where most collected data is unstructured and highly domain-specific. This leads us to transfer learning, which is a method that reuses a pre-trained model developed for a task and applies it to another task. The process of transfer learning is illustrated in figure 1.1.

Classification of transfer learning types

In recent years since the introduction of BERT (Bidirectional Encoder Representations from Transformers) [12] by researchers at Google AI Language in 2018, transfer learning in the form of pretrained language models has become ubiquitous in NLP and has achieved state-of-the-art results on a wide range of tasks. However, transfer learning is not recent phenomenon in NLP as lots of approaches for transfer learning have been studied since the mid 2000s. [29] There are different types of transfer learning common in current NLP. Differences between approaches are shown in figure 1.2. They are classified based on:

- Whether the source and target settings deal with the same task.
- The nature of the source and target domains.

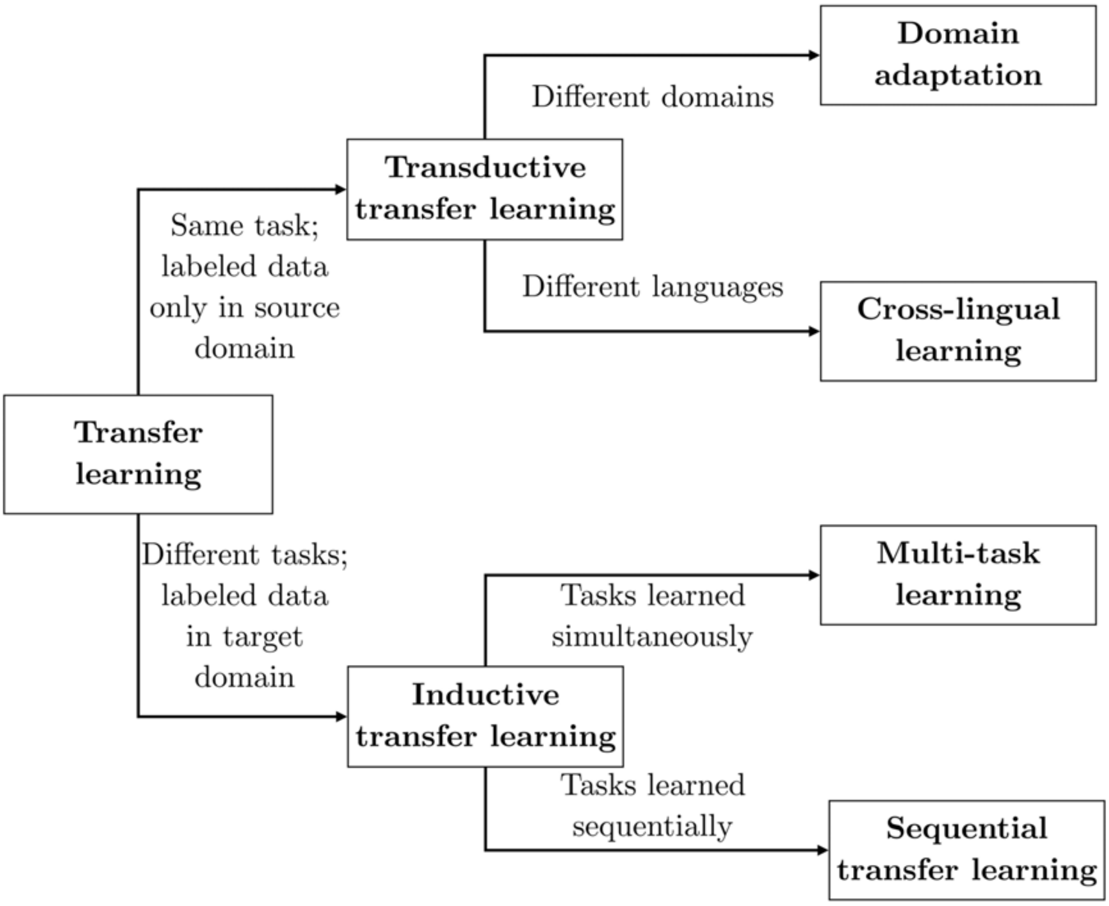

Domain adaptation
Different domains
Transductive transfer learning
Same task; labeled data only in source domain
Different languages
Cross-lingual learning
Transfer learning
Different tasks; labeled data in target domain
Multi-task learning
Tasks learned simultaneously
Inductive transfer learning
Tasks learned sequentially
Sequential transfer learning

Figure 1.2: Classification of transfer learning types [29].

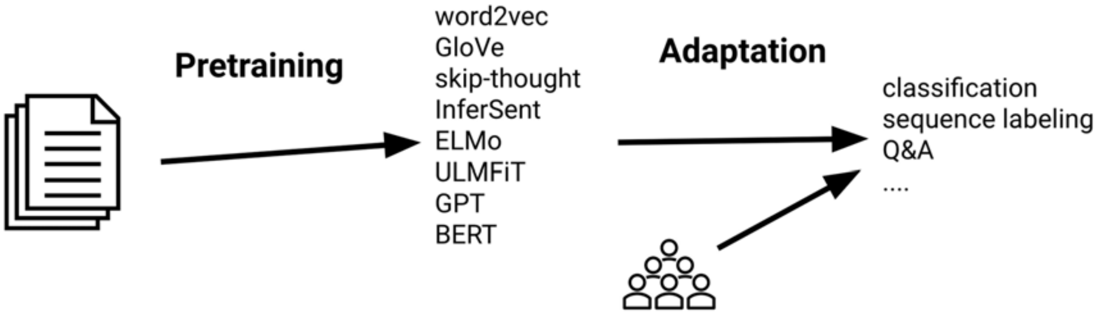


Figure 1.3: The general procedure of sequential transfer learning [29].

- The order in which the tasks are learned.

Sequential transfer learning

The biggest breakthroughs so far in the field of NLP use sequential transfer learning. With this type of learning, we first pre-train representations on large corpora of unlabelled domain-general text such as the entire Wikipedia, books, etc. using our method of choice and then to adapt these representations to a supervised target task such as classification, sequence labeling, or Q&A, etc. using a much smaller set of labeled data. The general procedure of sequential transfer learning can be found in figure 1.3.

Throughout our study, we will use be using these general pre-trained language models. This type of model is robust because it allows us to perform different tasks such as aspect-based sentiment analysis and text summarization without the need for labeled data or training our models and thus makes our system unsupervised and highly applicable to other domains.

1.3 Objectives

The study proposes a machine learning system for automatic summarization of opinions found in users' video games reviews on Steam. The objectives of the thesis are:

- To gather a large number of reviews from several different games on Steam for experiments and evaluation.
- To utilize and combine existing unsupervised methods to perform aspect-based sentiment analysis and text summarization of the reviews.
- To build a dashboard that enables beautiful visualization of the results and further analyses.

1.4 Structure

The study is split into six chapters:

- **Chapter 1** (this chapter) gives a brief introduction to the problem of opinion summarization of video game reviews and why transfer learning methods are important.

- **Chapter 2** presents the conceptual backgrounds for the core building blocks of our system, which are aspect-based sentiment analysis, text summarization, and data visualization. Related works and state-of-the-art methods are also discussed in comparison with our methods.
- **Chapter 3** gives an overview of our system, its core building blocks, their inspiration and motivation are explained. The process of gathering the data as well as a database model storing the reviews collected are presented.
- **Chapter 4** goes into details of our methods for doing aspect extraction, sentiment analysis, text summarization, and data visualization.
- **Chapter 5** presents the experiments and evaluation of our system.
- **Chapter 6** summarizes all the work done, discusses the potential of the proposed system, techniques, and future work that can be done on top of this study.

2 Conceptual backgrounds

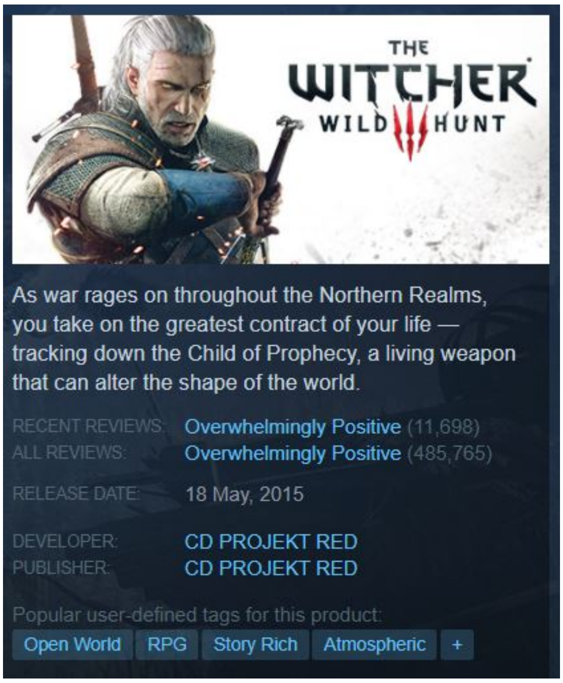
2.1 Opinion summarization

Opinions can bow be found practically everywhere from social networks like Facebook, Twitter to news portals, e-commerce sites, etc. thanks to the explosive growth of the web over the last two decades. While these opinions are supposed to be helpful, the vast availability of such opinions becomes overwhelming to users as there is just too much to digest. Consider a user looking to buy a game on Steam. Figure 2.1 shows an overview of the game *The Witcher 3: Wild Hunt*[1] on Steam. As one of the most successful video games of all time, the game accumulates nearly half a million reviews in 28 different languages on Steam ever since its release. Other more popular titles can have millions of reviews. Such overwhelming amounts of information make summarization of the web very critical.

Over the last decade, this special task of summarizing opinions has stirred tremendous interest amongst the NLP communities. Opinions mainly include opinionated text data such as reviews, tweets, Facebook posts, etc. Opinion summarization attempts to generate a concise and digestible summary of a large number of opinions. [19]

The simplest form of an opinion summary is the result of sentiment prediction, which is the task of identifying whether the underlying sentiment of a piece of text is positive, negative, or neutral, and then aggregating the sentiment scores. The sentiment can sometimes be given directly by the user without us having to predict. On the Steam platform, in addition to writing the review, the user is also asked to provide their overall feeling about the game: *Recommended* (i.e., a positive review), or *Not Recommended* (i.e., a negative review). In figure 2.1, as the number of positive reviews outweighs negative ones heavily, an overall sentiment of *Overwhelmingly Positive* for both recent

[1] https://store.steampowered.com/app/292030/The_Witcher_3_Wild_Hunt/



As war rages on throughout the Northern Realms, you take on the greatest contract of your life — tracking down the Child of Prophecy, a living weapon that can alter the shape of the world.

RECENT REVIEWS: Overwhelmingly Positive (11,698)
ALL REVIEWS: Overwhelmingly Positive (485,765)

RELEASE DATE: 18 May, 2015

DEVELOPER: CD PROJEKT RED
PUBLISHER: CD PROJEKT RED

Popular user-defined tags for this product:
Open World RPG Story Rich Atmospheric +

Figure 2.1: Overview of the game The Witcher 3: Wild Hunt on Steam.

and all reviews, which acts as the simplest form of summary for users' perceptions about the game, is shown.

Beyond such summaries, the newer generation of opinion summaries includes structured summaries that provide a well-organized breakdown by aspects, various formats of textual summaries, and temporal visualization. The different formats of summaries complement one another by providing different levels of understanding. [19] For example, sentiment prediction on reviews of a game can give a very general notion of what the users feel about the game. If the user needs more specifics, then the aspect-based summaries or textual summaries may be more useful. Regardless of the summary formats, the goal of opinion summarization is to help users digest the vast availability of opinions in an easy manner. The approaches utilized to address this task vary greatly and touch different intertwined areas of research including language modeling, text classification, information extraction, text summarization, topic modeling, data visualization, and so on. Some of these approaches rely on simple heuristics, while others use robust machine learning or deep learning models.

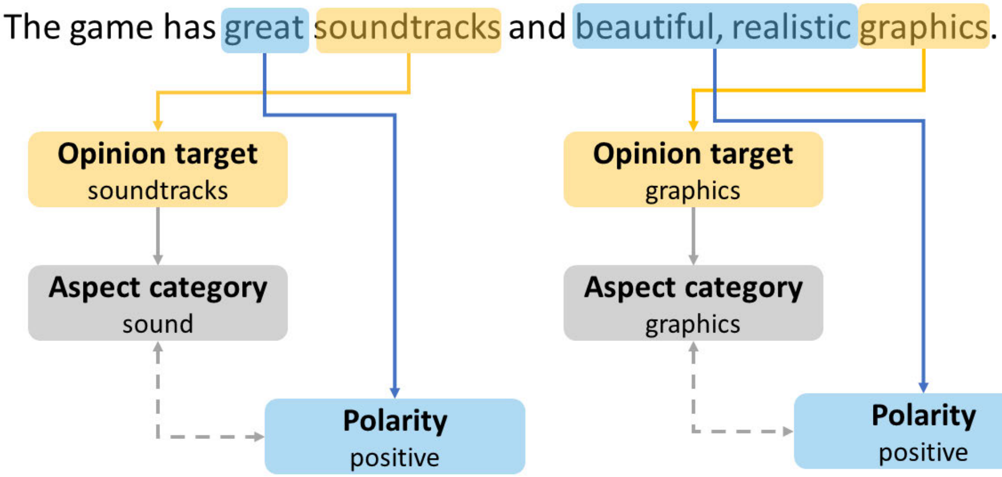


Figure 2.2: The 3 subtasks of aspect-based sentiment analysis in a sample sentence of a Stream review

In the following sections, we will go into detail about the 3 core related research areas used to build our opinion summarization system, which are aspect-based sentiment analysis, text summarization, and data visualization.

2.2 Aspect-based sentiment analysis

Rather than simply classifying user reviews as *positive*, *negative*, or *neutral*, over a period of time, sentiment analysis has evolved into a more sophisticated paradigm which is called aspect-based sentiment analysis. This task is quite new as it has just been formally defined since 2014 in SemEval[2] (International Workshop on Semantic Evaluation) [27]. Figure 2.2 illustrates the 3 subtasks of aspect-based sentiment analysis on a sample sentence of a Steam review. Those 3 subtasks are:

- **Opinion target extraction**: We extract the opinion target, also referred to as aspect term, from the sentence, in this case, *soundtracks*, *graphics*.
- **Aspect category detection**: Given some predefined categories, the task is to identify the categories of the aspect terms. The aspect of *soundtracks* is *sound* and the aspect of *graphics* is *graphics*.

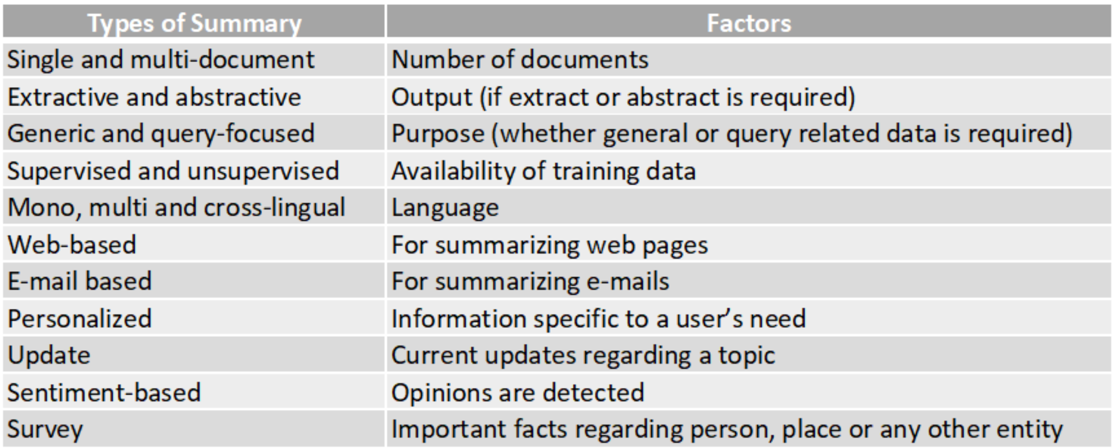
[2] https://semeval.github.io/

- **Sentiment polarity identification**: In this task, we try to identify the sentiment of an aspect term, either *positive* or *negative*. In our example, both aspect terms are identified as *positive*.

For this kind of task, one can simply fine-tune a pre-trained language model such as BERT using labeled training data to solve the 3 subtasks to obtain the triples (opinion target, aspect category, and polarity) as shown in figure 2.2. Recent results showed that BERT with fine-tuning achieved state-of-the-art performance in many extraction tasks, outperforming previous customized neural network approaches. [31] However, a significant amount of high-quality labeled training data is still required for this fine-tuning approach to work. Labeled training data can be obtained by using crowdsourcing platforms such as Amazon Mechanical Turk, MicroWorkers, ClickWorker, MicroTask, Scale, etc. to employ data labeling workers. But this also introduces new problems as preparing crowdsourcing task, launching and post-processing the results are usually very time-consuming. The process often needs to be repeated a few times to make necessary adjustments. Malicious crowdworkers is also a problem that needs to be addressed to ensure the quality of our data. Labels for a sentence must be collected several times to reduce possible errors and the results have to be cleaned before they are usable for downstream tasks. Even worse, this expensive labeling process must be repeated to train a new model from a different domain [31].

For our study, reviews from different genres of games or even different games often have really diverse sets of possible aspect categories. So trying to predefine commonly found aspects in video games such as *gameplay*, *story*, *sound*, *graphics*, *combat*, etc. for all games may not be the best idea as we will miss lots of aspects. Games always contain much more distinguished features and mechanics than other domains such as hotels, restaurants, smartphones, movies, etc. Even if we manage to create a high-quality labeled dataset for a set of games, there is little chance that the models trained on such dataset will work on unseen games.

Motivated by the aforementioned issues, we investigate the problem of extracting aspects and their corresponding sentences without the need for labeled data or predefined aspects. This leads us to use a simple rule-based approach to extract certain keywords from sentences. Then clustering method helps us group similar keywords together to form aspects. The sentiment classification, which is also unsupervised, is performed on all sentences in each aspect.



Types of Summary	Factors
Single and multi-document	Number of documents
Extractive and abstractive	Output (if extract or abstract is required)
Generic and query-focused	Purpose (whether general or query related data is required)
Supervised and unsupervised	Availability of training data
Mono, multi and cross-lingual	Language
Web-based	For summarizing web pages
E-mail based	For summarizing e-mails
Personalized	Information specific to a user's need
Update	Current updates regarding a topic
Sentiment-based	Opinions are detected
Survey	Important facts regarding person, place or any other entity

Figure 2.3: Different text summarization types based on various factors [16]

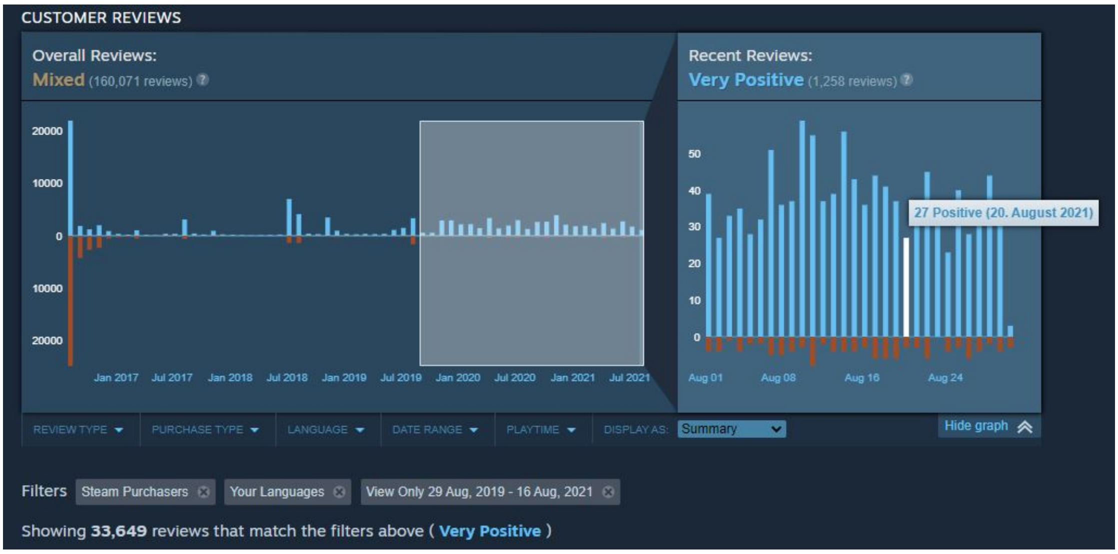
2.3 Text summarization

Text summarization refers to the task of creating a summary of a longer piece of text. The objective of this task is to create a coherent summary that captures the key ideas in the text [33]. It is useful to do a quick read of large documents, store only relevant information, and facilitate better retrieval of information at the end.

There are various types of text summarization, which are shown in figure 2.3 together with the factors determining summarization type. The important ones to understand are:

- **Single-document versus multi-document summarization**: In single-document summarization, the summary is generated from a single document, whereas in multi-document summarization, many documents are used for generating a summary. Redundancy is one of the biggest problems in summarizing multiple documents. [16]

- **Extractive and abstractive**: Extractive summarization creates a summary by selecting a few relevant sentences from the original document, whereas abstractive summarization uses words and phrases different from the ones occurring in the source document. Due to its simplicity and feasibility, extractive summarization has become a standard in text summarization, whereas abstractive summarization is more of a research topic than a practical application.



CUSTOMER REVIEWS

Overall Reviews:
Mixed (160,071 reviews)

Recent Reviews:
Very Positive (1,258 reviews)

DISPLAY AS: Summary

Filters Steam Purchasers Your Languages View Only 29 Aug, 2019 - 16 Aug, 2021

Showing **33,649** reviews that match the filters above (**Very Positive**)

Figure 2.4: Steam's customer reviews data visualization

- **Generic and query-focused**: Query-focused summarization refers to creating the summary of the text depending on the user query, whereas generic summarization creates a general summary.

In our study, once we derive all the aspects and tag specific sentences with them from the previous step, it is possible to group the sentences by aspects. Given the huge volume of video game reviews on Steam, there will still be lots of sentences under an aspect. We solve this problem by using clustering methods to pick and choose the most salient sentences for each aspect. And we intend to generate summaries based only on certain aspects that the users are interested in, which means that the type of text summarization we perform is multi-document, extractive, query-based, and unsupervised summarization.

2.4 Data visualization

Data visualization is the presentation of data in a pictorial or graphical format. [30] Seeing analytics presented visually makes it easier for decision-makers to understand difficult concepts or discover new patterns. Beyond static charts and graphs, interactivity can be utilized to help us drill down into visuals for more details and dynamically change what data we see and how it is processed. This greatly enhances user experience as it not only allows a much greater amount of data to be presented in space but also facilitates more in-depth and complex analytics comparing to static visualization.

Data visualization is important because of the way the human brain processes information. It is much simpler to visualize large amounts of complex data using charts or graphs instead of poring over spreadsheets or reports. Data visualization is a quick, easy way to convey concepts in a universal manner – and we can experiment with different scenarios by making slight adjustments.

Here is how data visualization can help us make sense of video game reviews [30]:

- **Comprehend reviews quickly**: By using graphical representations of reviews information, we are able to see large amounts of data in clear, cohesive ways – and draw conclusions from that information. And since it is significantly faster to analyze reviews in graphical format (as opposed to reading through random reviews in their plain text format), we can address problems or answer questions in a more timely manner.

- **Pinpoint emerging trends**: Using data visualization to discover trends – both in our own game or in the market – can give us an edge over the competition, and ultimately affect the bottom line. It is easy to spot outliers that affect the game quality and address issues before they become bigger problems.

- **Identify relationships and patterns**: Even extensive amounts of complicated reviews start to make sense when presented graphically. We can recognize parameters that are highly correlated. Some of the correlations will be obvious, but others will not. Identifying those relationships helps us focus on areas most likely to influence our most important goals.

- **Communicate the story to others**: Once we have uncovered new insights from visual analytics, the next step is to communicate those insights to others. Using charts, graphs or other visually impactful representations of data is important in this step because it is engaging and gets the message across quickly.

The Steam platform does provide us with a simple interactive visualization of customer reviews as seen in figure 2.4. Two vertical bar charts show the number of positive and negative reviews of all and also recent (last 30 days) reviews respectively. A time slider can be used to focus on a specific timespan. We can also tweak lots of different filters such as *review type*, *purchase type*, *language*, *playtime*, etc., or click directly on the graphs to extract a subset of reviews from the whole. Our study takes this visualization a step further and introduces new elements such as aspect-based sentiment analysis, text summarization, mentions analysis, etc.

2.5 Related work

The importance of analyzing user reviews has drawn a great deal of interest among researchers. There have been a plethora of studies utilizing various techniques for opinion summarization of user reviews from different domains such as Amazon product reviews, IMDB movie reviews, hotel, and restaurant reviews, etc.

Despite the widespread appeal of video games, there has been little discussion on this particular domain. [37] proposes an aspect-based summarization system for Steam reviews. They employ a modified double propagation (DP) algorithm for extracting aspect-sentiment word pairs. Following this, they use a seed list and word similarity to categorize aspect terms into groups, thus producing an aspect-based summary. [25] tries to solve the problem of video game reviews summarization by applying k-means clustering in order to identify groups of similar sentences. They then utilize word lists with the aim of mapping the produced clusters to predefined game aspects, like *graphics*, *gameplay*. Subsequently, they apply sentiment analysis using a rule-based method in order to extract the sentiments that pervade each cluster. As discussed in previous sections, the results of these studies are not so good because they rely heavily on these seed list and predefined aspects. And as the number of topics discussed in video games is so vast and different, it is infeasible to use only k-means and seed list to extract all the aspects from the reviews.

One of the simpler approaches to the task of opinion summarization is general-purpose summarization [7, 24], which tries to optimize the salience of the overall generated content. Aspect-based summarization is more challenging as it is aspect-centric and focuses on maximizing the diversity of opinions being discovered in the final summary. Supervised methods for opinion summarization such as [32] depend on large annotated dataset of document-summary pairs to train their models, which makes them difficult to adapt across different domains [5]. Motivated by recent works such as [3, 38], we propose an unsupervised extractive summarization framework built on top of an unsupervised aspect extraction module and unsupervised sentiment analysis module, which makes our approach applicable to other domains.

Video game reviews are posted online to capture a wide range of human emotions owing to reviewers' social and cultural backgrounds. State-of-the-art methods for opinion summarization [5, 1] do not allow for customizable while generating summaries from such a diverse range of opinionated text. We, however, argue that readers should be able to customize the shape and content of such summaries to suit their varying interests. To our knowledge, only [2, 15] have motivated the need for controllable summarization. However, both of them are supervised techniques and require gold-standard summaries for training their models. Our proposed framework is unsupervised and extractive in nature, which additionally lets our readers control several attributes of the summary such as its length, specific aspects of interest it must focus on, timespan, sentiments, etc.

Many studies just stop at producing a framework for opinion summarization. Our study goes one step further and tries to build a visualization tool for the results generated. Many commercial or academic summary visualization systems [17, 21] have been built for generating and exploring summaries. While these summary visualization systems are effective for reducing redundancy that occurs in reviews, there are two major limitations. First, the generated summaries are mostly static and the granularity of the summary cannot be tailored to user needs. For example, summaries that are generated by aggregating positive and/or negative sentiments over *sound* may be too general for users who are only interested in information about *voice acting*. At the same time, it is also too detailed for users who simply wish to know the overall aggregated rating. Second, most existing summarization systems cannot explain the summary that is generated or allow users to explore different aspects of the generated summary. To address these limitations, we develop a visualization prototype that can:

- Generate extractive summaries according to user's preferences.
- Enable users to interactively explore different aspects, keywords, sentiments, etc. by defining their own level of desired granularity.

3 Data and system overview

3.1 The Steam platform

Steam is a digital game distribution platform, developed by Valve Corporation. Steam is the largest digital distribution platform for PC gaming, with over 10,000 games [10] released on the site and over 120 million monthly active players in 2020 [9]. Steam offers digital rights management (DRM), multiplayer gaming, and social networking services, through two major components of the Steam platform: the Steam Store [1], and the Steam Community [2].

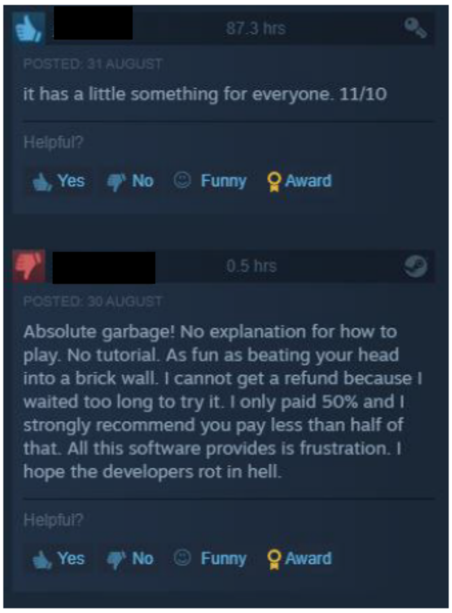
Games can be purchased directly from the Steam Store or from third-party vendors and then activated through the Steam platform, which are playable for a user after logging in on Steam using the Steam client. The Steam client will verify ownership of the game and any available updates will automatically be installed. Install the latest update is required in order to play a game through Steam.

In addition, Steam Community offers users social network-like features such as friend lists. Game developers and journalists can publish news updates for games on so-called channels. In general, although it is not compulsory for developers to post announcements about game updates to one or more channels (e.g., to the Product Update channel), developers often do post news updates about their games to keep users informed about the latest news about their games. [20]

The Steam Community also permits users to post reviews of games once they played them. Different from other popular application distribution platforms which use a 5-star rating system for reviews, players are asked to provide their overall feeling about the game: *Recommended* (i.e., a positive review), or *Not Recommended* (i.e., a negative review). The number of playing hours of the reviewed game, the number of played games,

[1] https://store.steampowered.com/

[2] https://steamcommunity.com/



[redacted] 87.3 hrs

POSTED: 31 AUGUST

it has a little something for everyone. 11/10

Helpful?

[redacted] 0.5 hrs

POSTED: 30 AUGUST

Absolute garbage! No explanation for how to play. No tutorial. As fun as beating your head into a brick wall. I cannot get a refund because I waited too long to try it. I only paid 50% and I strongly recommend you pay less than half of that. All this software provides is frustration. I hope the developers rot in hell.

Helpful?

Figure 3.1: A positive and a negative review on Steam

and the number of previously posted reviews by the reviewer at this moment are shown alongside the review. The positive review rate is displayed on the Steam Store page of the game, to advise potential customers. A user can only provide one review for each game, across all versions of the game, and updating the review at a later time is possible. A review can also be marked as *Helpful*, *Funny*, etc. by other community members. An example of a positive and a negative review on Steam can be seen in figure 3.1.

3.2 Reviews scraping

In this study, we use a modified version of a Steam reviews scraping package called steamreviews[3]. The library makes use of the Steam API[4] in order to download every Steam review for the games of our choice. However, some games can exceed hundreds of thousands, even millions of reviews as they are very popular or have been around for a long time. So we customize the package so that the number of reviews we want to collect can be specified before scraping. For our study, we collect a maximum of 5000 newest reviews in English for each game.

[3] https://pypi.org/project/steamreviews/

[4] https://partner.steamgames.com/doc/store/getreviews

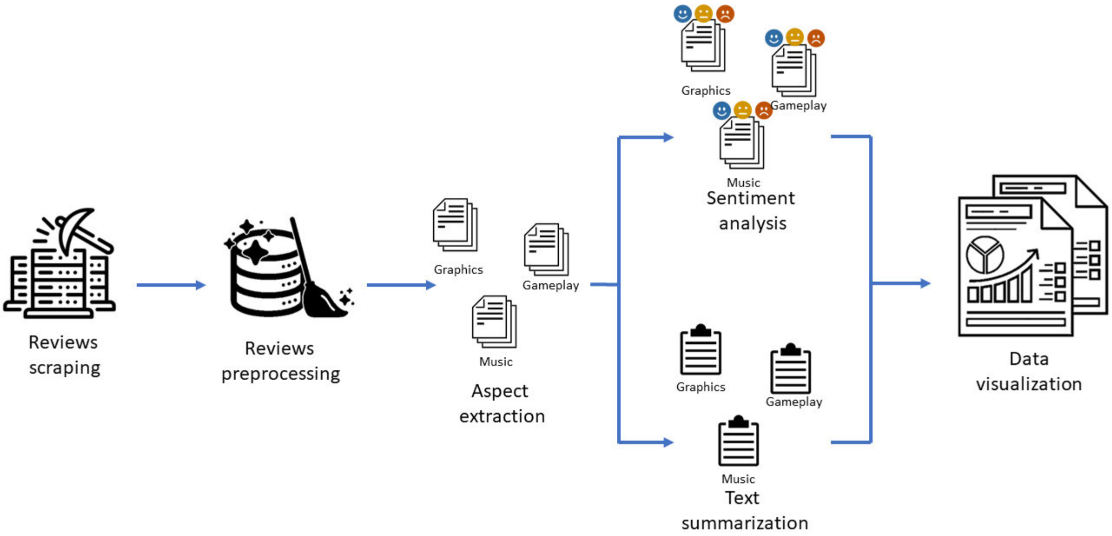


Figure 3.2: System overview

Reviews in English accounts for nearly 50% of the total number of reviews for every game. In this study, we choose to focus only on English reviews as they are available in abundance compared to other languages and most of the pre-trained models we use are trained using English data. When posting a review, the user has to provide the language the review is written in. As this is not checked for by the system, a few reviews in unwanted languages can still slip in, which must be addressed later during our preprocessing step.

3.3 System overview

This section gives an overview of the entire system and formally defines the problem. We will go into a more in-depth discussion of each component in chapter 4.

The pipeline of our system in Figure 3.2 works as follows:

- **Reviews scraping**: Collect the reviews of a game using the reviews scraper.
- **Reviews preprocessing**: Preprocess the reviews and split each of them into sentences. These sentences also need to be cleansed and filtered as some contain inappropriate symbols or structures which will decrease the effectiveness of our system in the later steps.

- **Aspect extraction**: Use a heuristic-based approach that utilizes part-of-speech (POS) tagging and certain rules to extract keywords (or keyphrases). We then compute the sentence embeddings of each extracted keyword using SBERT (Sentence-BERT) [28], and apply agglomerative clustering to cluster similar ones together into groups, which are called aspects. Keywords are also extracted together with the sentences in which they are found so these sentences can be used in the subsequent steps.

The two following components come after aspect extraction and can be performed simultaneously:

- **Sentiment analysis**: Use a pre-trained RNN-based sentiment classifier to classify the sentences found in the previous step as *positive*, *negative*, or *neutral*.
- **Text summarization**: For each aspect, apply SBERT to compute the sentence embeddings and use a combination of HDBSCAN and agglomerative clustering to cluster sentences. We then find the sentence lying closest to the center of each cluster and call it an opinion. We rank the opinions by the size of their clusters and pick out the 5 most popular opinions. We assume that the bigger the cluster, the more popular an opinion is. And thus for each extracted aspect from the previous step, we have a list of the 5 most important opinions, which will act as a summary for that aspect.

The last step utilizes the output from the previous steps, which is saved into an SQL database model.

- **Data visualization**: Connect Power BI to our database model and put together interactive dashboards that help us visualize and perform further complex analysis with time, aspects, keywords, etc.

3.4 Database model

A good database design is crucial in ensuring consistent data, elimination of data redundancy, efficient execution of queries, and high-performance application. Taking the time to design a database saves time and frustration during development, and a well-designed database ensures ease of access and retrieval of information.

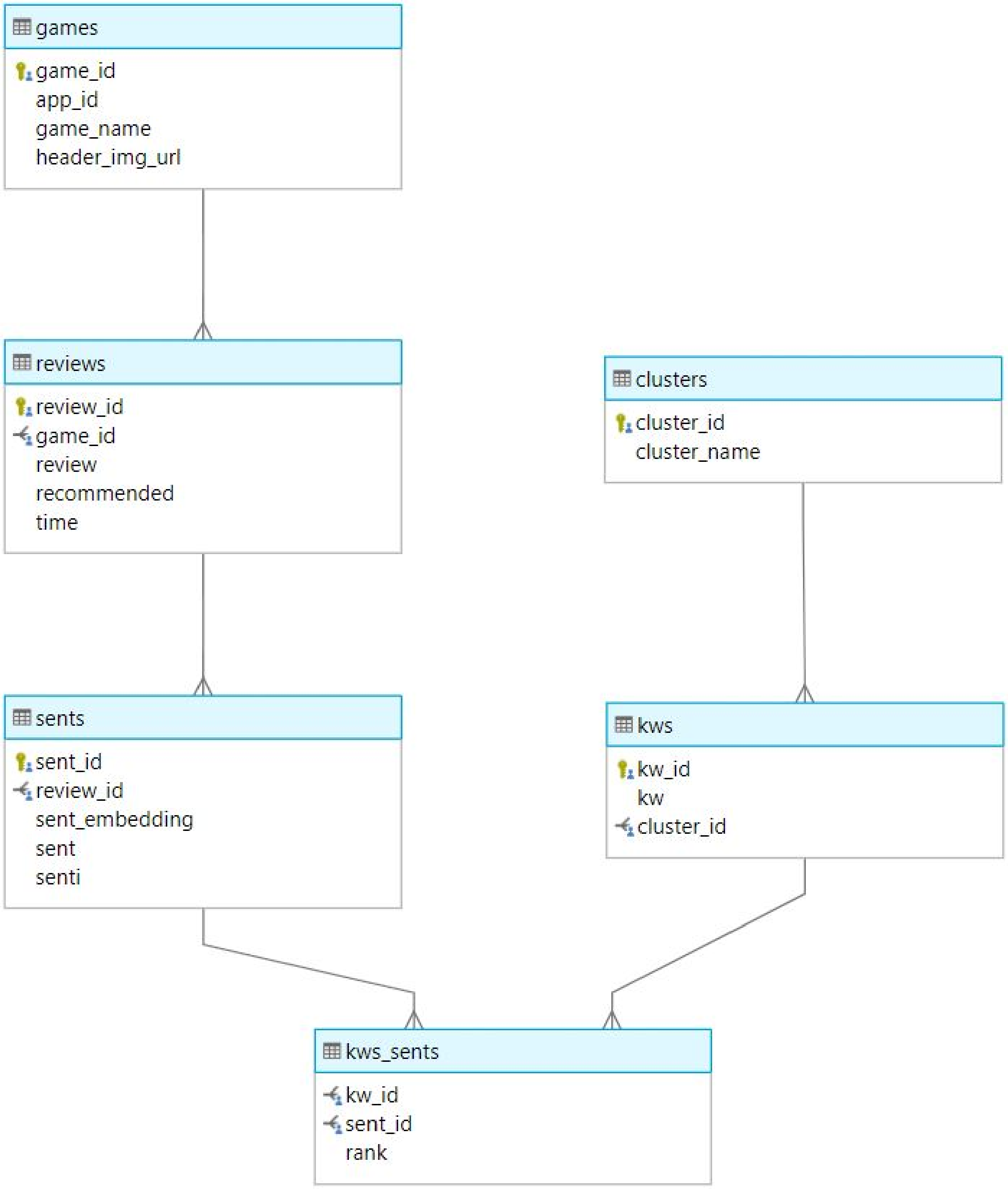
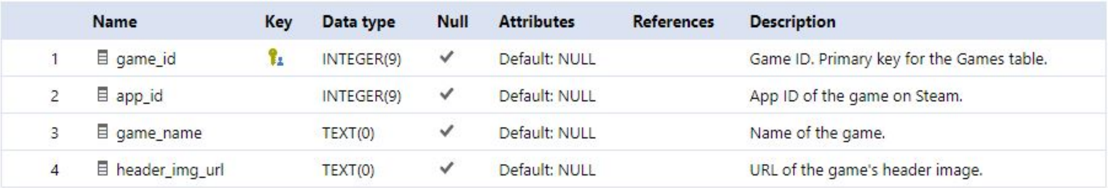
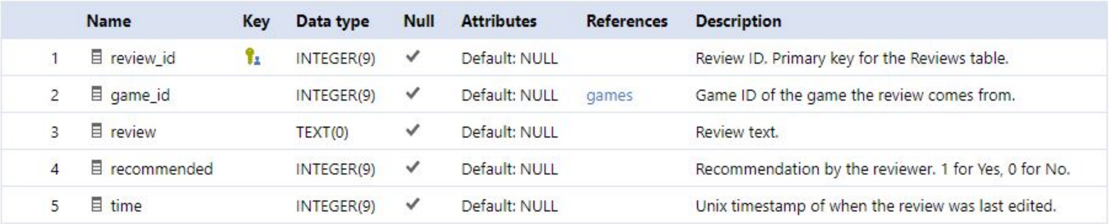


Figure 3.3: ER Diagram of our system



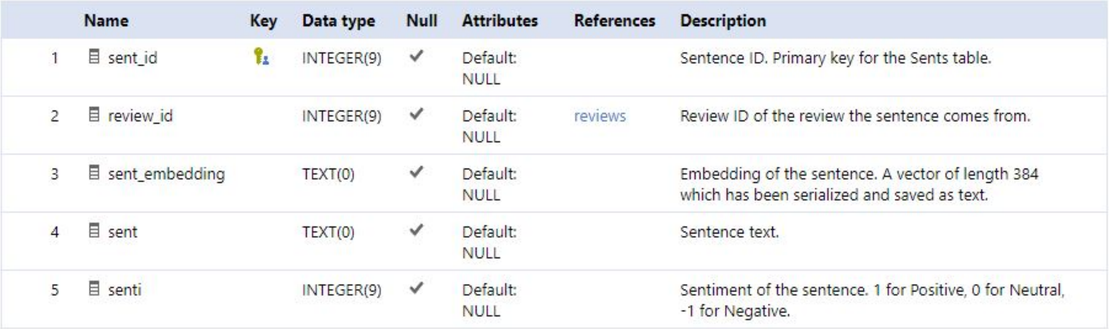
	Name	Key	Data type	Null	Attributes	References	Description
1	game_id	🔑	INTEGER(9)	✓	Default: NULL		Game ID. Primary key for the Games table.
2	app_id		INTEGER(9)	✓	Default: NULL		App ID of the game on Steam.
3	game_name		TEXT(0)	✓	Default: NULL		Name of the game.
4	header_img_url		TEXT(0)	✓	Default: NULL		URL of the game's header image.

Figure 3.4: Table GAMES: Games in our system.



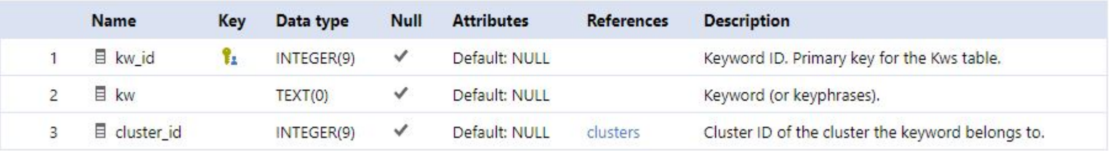
	Name	Key	Data type	Null	Attributes	References	Description
1	review_id	🔑	INTEGER(9)	✓	Default: NULL		Review ID. Primary key for the Reviews table.
2	game_id		INTEGER(9)	✓	Default: NULL	games	Game ID of the game the review comes from.
3	review		TEXT(0)	✓	Default: NULL		Review text.
4	recommended		INTEGER(9)	✓	Default: NULL		Recommendation by the reviewer. 1 for Yes, 0 for No.
5	time		INTEGER(9)	✓	Default: NULL		Unix timestamp of when the review was last edited.

Figure 3.5: Table REVIEWS: Reviews of each game. A review can belong to only one game and have multiple sentences.



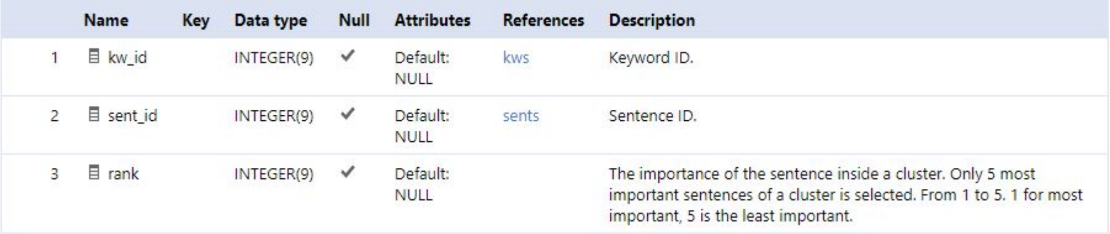
	Name	Key	Data type	Null	Attributes	References	Description
1	sent_id	🔑	INTEGER(9)	✓	Default: NULL		Sentence ID. Primary key for the Sents table.
2	review_id		INTEGER(9)	✓	Default: NULL	reviews	Review ID of the review the sentence comes from.
3	sent_embedding		TEXT(0)	✓	Default: NULL		Embedding of the sentence. A vector of length 384 which has been serialized and saved as text.
4	sent		TEXT(0)	✓	Default: NULL		Sentence text.
5	senti		INTEGER(9)	✓	Default: NULL		Sentiment of the sentence. 1 for Positive, 0 for Neutral, -1 for Negative.

Figure 3.6: Table SENTS: Sentences of each review. A sentence can belong to only one review and have multiple keywords.



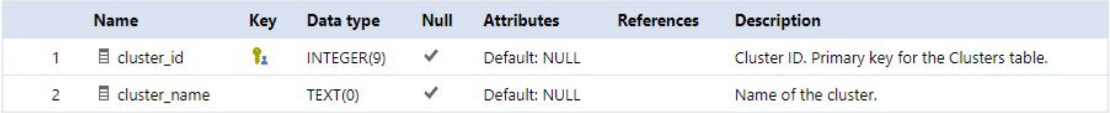
	Name	Key	Data type	Null	Attributes	References	Description
1	kw_id	🔑	INTEGER(9)	✓	Default: NULL		Keyword ID. Primary key for the Kws table.
2	kw		TEXT(0)	✓	Default: NULL		Keyword (or keyphrases).
3	cluster_id		INTEGER(9)	✓	Default: NULL	clusters	Cluster ID of the cluster the keyword belongs to.

Figure 3.7: Table KWS: Keywords (or keyphrases). A keyword can belong to only one cluster, but multiple sentences.



	Name	Key	Data type	Null	Attributes	References	Description
1	kw_id		INTEGER(9)	✓	Default: NULL	kws	Keyword ID.
2	sent_id		INTEGER(9)	✓	Default: NULL	sents	Sentence ID.
3	rank		INTEGER(9)	✓	Default: NULL		The importance of the sentence inside a cluster. Only 5 most important sentences of a cluster is selected. From 1 to 5. 1 for most important, 5 is the least important.

Figure 3.8: Table KWS_SENTS: A bridge table to model the many-to-many relationship between sentences and keywords.



	Name	Key	Data type	Null	Attributes	References	Description
1	cluster_id	🔑	INTEGER(9)	✓	Default: NULL		Cluster ID. Primary key for the Clusters table.
2	cluster_name		TEXT(0)	✓	Default: NULL		Name of the cluster.

Figure 3.9: Table CLUSTERS: Clusters (or aspects) of keywords. A cluster can have multiple keywords.

The database model is a vital component in our machine learning system. We collect, store and update the data in our database at every aforementioned step. For creating our database, we use SQLite, a popular SQL database engine. Figure 3.3 shows the entity-relationship diagram of our entire system. Detailed definition of each table can be found in figures 3.4, 3.5, 3.6, 3.7, 3.8, 3.9.

4 Methods

4.1 Review preprocessing

It is common knowledge that words that appear in reviews often have many structural variants. Thus, before any NLP task can be applied to the reviews, data preprocessing techniques must be utilized. Such preprocessing techniques are essential in order to convert our initial reviews into a suitable format which will increase the effectiveness of any NLP system. [25] In our approach, we build a pipeline to preprocess the reviews, split them into sentences, and apply a set of cleansing operations to make sure each sentence that goes into the next steps is as structured and meaningful as possible. Figure 4.1 shows the entire pipeline.

There are 2 sets of operations. The first set of operations is used on the original reviews that come directly from Steam. The reviews need to be cleaned extensively so that the sentence boundary detection of the sentence tokenization can work properly:

- **Remove short reviews**: Steam reviews have a length requirement of only one character so there are lots of short and meaningless reviews such as *"Yes."*, *"Great."*, *"Good game."*, *"Bad."*, *"!"*, etc. which contribute nothing to our aspect-based sentiment analysis or text summarization. There are also spam reviews such as *"So goooooooooooooooooooooooood!!"*, *"Good good good good good good good good good"*, etc. We remove any review where the number of unique tokens is less than 3.
- **Remove Steam markup tags**: Steam allows the users to use markup tags to add formatting to the text of the reviews. An example of a couple of tags can be found in figure 4.2. These tags do not add any value to the data and need to be removed.

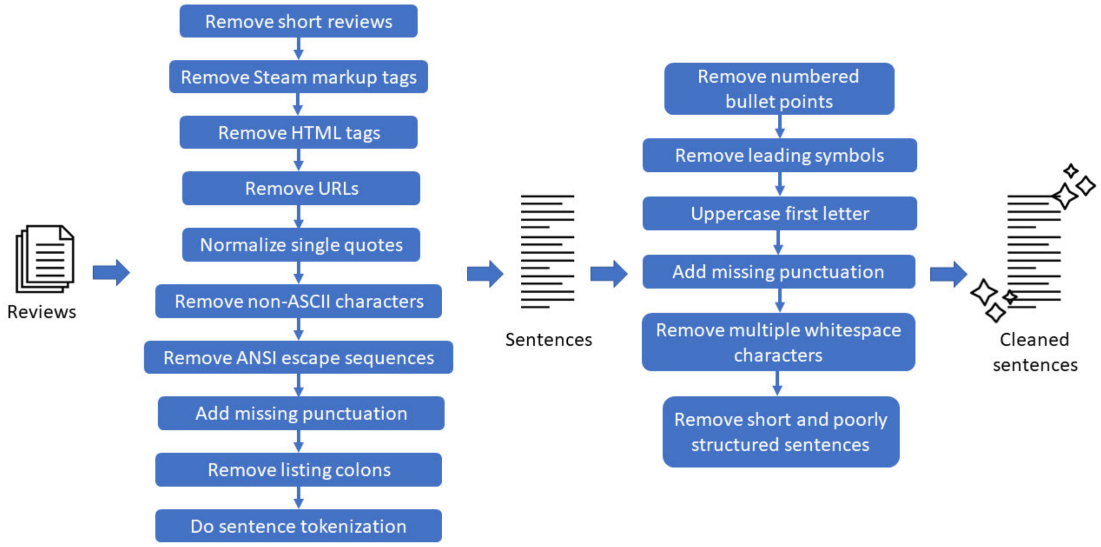
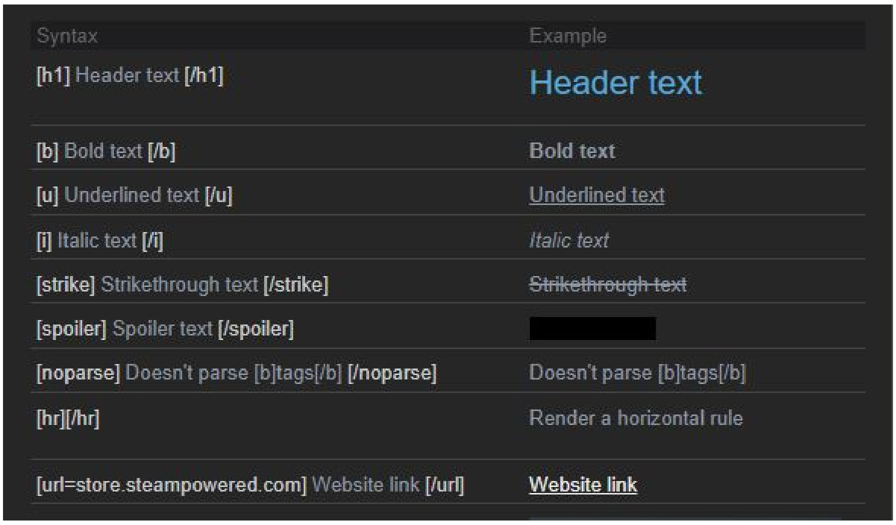


Figure 4.1: Reviews preprocessing pipeline



Syntax	Example
[h1] Header text [/h1]	Header text
[b] Bold text [/b]	**Bold text**
[u] Underlined text [/u]	Underlined text
[i] Italic text [/i]	*Italic text*
[strike] Strikethrough text [/strike]	~~Strikethrough text~~
[spoiler] Spoiler text [/spoiler]	■
[noparse] Doesn't parse [b]tags[/b] [/noparse]	Doesn't parse [b]tags[/b]
[hr][/hr]	Render a horizontal rule
[url=store.steampowered.com] Website link [/url]	Website link

Figure 4.2: Some available markup tags in Steam

- **Remove HTML tags**: HTML tags sometimes appear in the reviews. These tags do not add any value to the data and only enable proper browser rendering. We remove them all.
- **Remove URLs**: URLs (Uniform Resource Locators) are references to a location on the web, but do not provide any additional information. We remove them all.
- **Normalize single quotes**: The single quote (') can sometimes be found in non-ASCII versions in the reviews and cannot be interpreted by most of our models in the later steps. Single quotes are important than double quotes because they appear in lots of negative contractions such as *don't*, *doesn't*, *won't*, *wouldn't*, etc. which our models must interpret. So we replace them with the normal ones.
- **Remove non-ASCII characters**: As mentioned in section 3.2, reviews in languages other than English can sometimes slip in. Languages that use non-ASCII characters such as Thai, Chinese, Arabic, etc. will be removed cleanly in this step. Whereas languages using all or mostly ASCII characters such as Spanish, German, etc. can still be there in our data. But these cases are rare and acceptable. Non-ASCII characters such as accented, musical characters, mathematical characters, etc. cannot be interpreted by most of our models so we remove them.
- **Remove ANSI escape sequences**: We remove all ANSI escape sequences as they contribute nothing to our data.
- **Add missing punctuation**: Users do not always put a period at the end of their reviews. This greatly affects sentence tokenization as it depends on punctuation for sentence boundary detection. We put a period at the end of a review in case it is missing.
- **Remove listing colons**: Users sometimes use the colon together with a new line to start listing things. The sentence tokenization cannot detect the first item in the list as a separate sentence. So we replace the colon and the newline character with a period. This helps the model detect the sentence boundary better.
- **Do sentence tokenization**: For splitting a review in sentences, we use spaCy[1]'s `en_core_web_lg`[2] with only `tok2vec` and `parser` enabled to perform the tokenization faster. A transformer model `en_core_web_trf`[3] can also be used for

[1] https://spacy.io/

[2] https://spacy.io/models/en#en_core_web_lg

[3] https://spacy.io/models/en#en_core_web_trf

a small gain in accuracy, but the dependency parsing is so slow compared to the other model that it is not worth it.

The second set of operations is used to clean up the sentences. This is crucial as some sentences are still poorly structured, which will decrease the effectiveness of our system especially during sentiment analysis and text summarization:

- **Remove numbered bullet points**: Reviewers sometimes use numbered bullet points to list things out. These sentences starting with numbered bullet points can create confusion for users when a sentence is shown with a random numbered bullet point during summary and also decrease the accuracy of aspect extraction. So we must remove them.
- **Remove leading symbols**: A sentence can sometimes start with special characters such as $+$, -, $>$, etc. These are the results when reviewers try to list things out. We remove these unnecessary leading symbols.
- **Uppercase first letter**: The sentence tokenization does not always split a sentence at period. It is more robust and knows when to split sentences at conjunctions or other types of punctuation. This causes some sentences not to start with an uppercase letter. Also when trying to list things out, reviewers do not always capitalize the first letter of each bullet point. Some reviewers do not even care about capitalization. We capitalize the first letter of a sentence.
- **Remove multiple whitespace characters**: Commonly found whitespace characters such as the space (\s), the tab (\t), the new line (\n) and the carriage return (\r) can appear consecutively multiple times in a sentence and create unnecessary whitespaces. We replace them with a single whitespace.
- **Remove short and poorly structured sentences**: At the last preprocessing step, to make sure that most of the sentences which go into our system are meaningful and nicely structured, we remove all sentences that have less than 3 unique tokens or an uneven amount of brackets, double quotes as these tend to be mistakes during sentence tokenization and make the sentences hard to read or interpret.

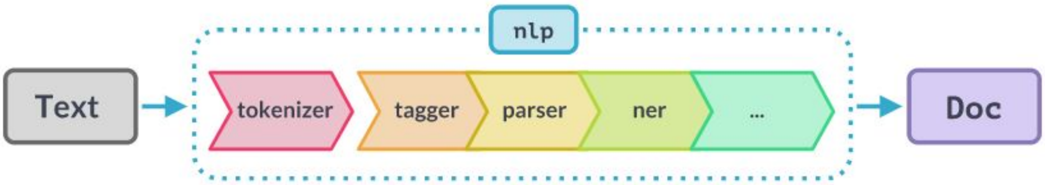
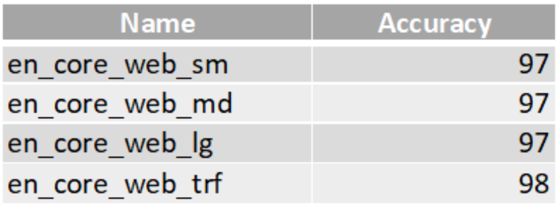


Figure 4.3: Language processing pipeline in spaCy



Name	Accuracy
en_core_web_sm	97
en_core_web_md	97
en_core_web_lg	97
en_core_web_trf	98

Figure 4.4: POS tagging accuracy (%) of trained pipelines for English in spaCy

4.2 Aspect extraction

In this step, we attempt to extract the aspects of a game that are mainly discussed by the reviewers. This can be quite challenging as video game aspects can be either explicitly or implicitly mentioned in a review text. For example, the sentence *"Easily my favorite game with realistic graphics."* clearly expresses an opinion about the aspect *graphics*. On the contrary, the sentence *"The grenade explosions look so fake."* does not mention the word *graphics* but it obviously refers to the graphics of the game, or possibly the physics engine. [25] Reviewers can sometimes

In our study, we use a heuristic-based approach that uses POS tagging and certain rules to extract keywords or keyphrases. The difference between keywords and keyphrases is that keywords are single words, while keyphrases are made up of a few words. But throughout this paper, we will refer to both of them as keywords. We then represent each keyword using SBERT and apply agglomerative clustering to cluster similar ones together into groups, which are called aspects. Keywords are also extracted together with the sentences in which they are found so these sentences can be used in the subsequent steps.

4.2.1 Keyword extraction

POS tagging

For this task, we employ spaCy[4], a free and powerful open-source library for natural language processing in Python. In order to perform POS tagging, spaCy needs a trained pipeline that supports a tagger. spaCy first tokenizes the text to produce a `Doc` object. The `Doc` is then processed in a processing pipeline. A typical language processing pipeline in spaCy can be seen in figure 4.3. It normally includes a tagger, a lemmatizer, a parser, and an entity recognizer. A trained component includes binary data that is produced by showing a system enough examples for it to make predictions that generalize across the language – for example, a word following "the" in English is most likely a noun. [14] The trained pipeline and its statistical models enable spaCy to make predictions of which tag most likely applies in this context.

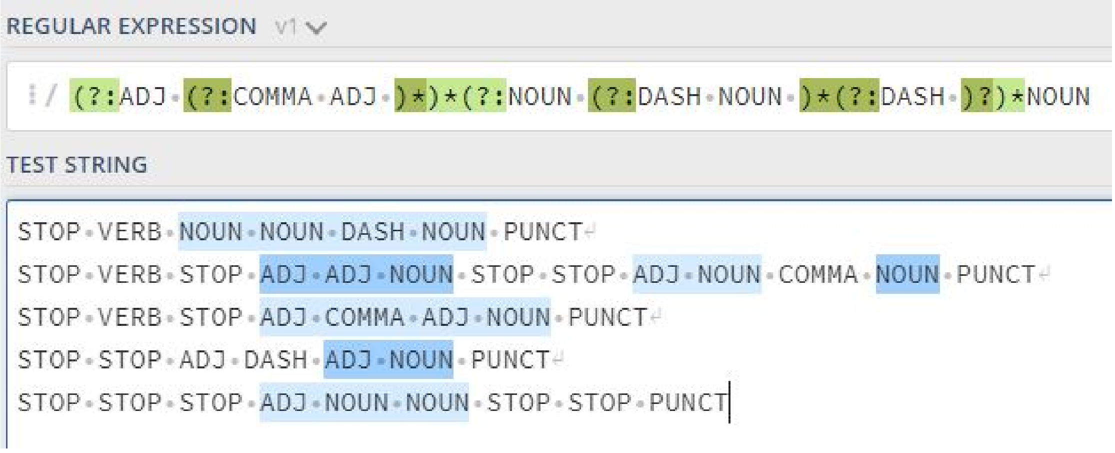
In spaCy, there are several trained pipelines available for English. The more complicated they are, the more accurate they can perform a task, but they are also a lot less efficient than the simpler ones. The accuracy of different trained pipelines when doing POS tagging can be found in figure 4.4. `en_core_web_trf`[5] is an English transformer pipeline (RoBERTa-based) [22]. Even though the difference is only 1% more in comparison to other pipelines, the transformer-based pipeline actually makes a big difference in accuracy when performing POS tagging with the preprocessed sentences in our study. When the grammar structures are complicated, which often are in the case of informal text from users, the pipeline correctly identifies the correct POS tags where others fail.

Keyword rules

In essence, we try to extract noun (or pronoun) chunks from a sentence. We define a noun chunk as a noun plus the adjectives that come before and describe them. For example, *big blue stone* or *amazing addicting gameplay*, etc. An attempt to define noun chunks more broadly to include more noun phrases like *a lot of games* (*of* is included), *good game* from the sentence *the game is good*, etc. has not been successful as the grammar structure is just too complicated to be defined by just using POS tagging. In theory, this approach can work with the help of dependency parsing and predefined rules. But they also introduce new problems as dependency parsing is much more computationally expensive than POS tagging and finding the rules that can capture such complexity of

[4] https://spacy.io/

[5] https://spacy.io/models/en#en_core_web_trf



```
/ (?:ADJ (?:COMMA ADJ )*)*(?:NOUN (?:DASH NOUN )*(?:DASH )?)*NOUN
```

```
STOP VERB NOUN NOUN DASH NOUN PUNCT
STOP VERB STOP ADJ ADJ NOUN STOP STOP ADJ NOUN COMMA NOUN PUNCT
STOP VERB STOP ADJ COMMA ADJ NOUN PUNCT
STOP STOP ADJ DASH ADJ NOUN PUNCT
STOP STOP STOP ADJ NOUN NOUN STOP STOP PUNCT
```

Figure 4.5: Regular expression for extracting noun chunks from POS tags strings

natural language, especially informal ones like our user reviews can prove to be a daunting task.

As simple as the definition of our noun chunks, extracting them from a sentence is still a non-trivial task. Some potential cases when trying to extract the noun chunks are:

- Adjectives can appear consecutively or be separated by commas.
- Nouns can appear consecutively or linked together by dashes.

spaCy's POS tagger tags all kinds of punctuation such as periods, commas, hyphens, quotes as *PUNCT* (for punctuation), which can be confusing when we are looking for specific patterns that include commas and dashes. Secondly, stop words can also appear as nouns or adjectives in a sentence. Removing these stop words helps us improve the performance as there are fewer and only meaningful noun chunks left. We solve this by introducing 3 new tags, which are *COMMA* (for commas), *HYPHEN* (for hyphens), and *STOP* (for stop words). These help us define a simpler pattern when looking for noun chunks. After POS tagging, we join the tags together and use the regular expression in figure 4.5 to extract the noun chunks.

The tags are then mapped to their respective tokens to form the noun chunks. The noun chunks are added together with the sentence ID (`sent_id`) of the sentence, from which they come from. If a noun chunk appears multiple times in a sentence, the count is still one.

There are lots of noun chunks that appear only once. There tends to be mispronunciation or things that are hardly mentioned by other reviewers. We remove them as they make up a big part of the noun chunks vocabulary and can slow down the performance of our system in subsequent steps.

4.2.2 Keywords embedding

After extracting the keywords, we notice that many of them actually refer to the same thing. For example, *music*, *soundtrack*, *great music*, *great soundtrack*, *sound*, *sound design*, *sound effects*, etc. all belong to the aspect *music*. Keywords such as *of graphics*, *animations*, *pixel art*, *animation*, *visuals*, etc. can all be classified under the aspect *graphics*. In other cases, there can be small misspellings in the keywords. The keyword *dark souls* is mentioned a lot in the reviews of *Momodora: Reverie Under the Moonlight*[6] as the game series *Dark Souls*[7] has a great influence on this title. But sometimes the name of the game is misspelled as *dark soul* without an *s* at the end. In our study, we try to group these similar keywords together using clustering algorithms.

But before clustering algorithms can be applied to the keywords, it is essential to convert them into real-valued vectors. A basic approach to this task is utilizing word embedding methods like word2vec [23] or GloVe (Global Vectors for Word Representation) [26] to compute the vector representation of each word in a keyword and average them to get the keyword embedding. This approach works with simple cases where the difference in spelling and and meaning is small(for example *bosses* and *boss*, *control* and *controls*. The problem is that word2vec and GloVe offer no context. So there is no difference between a *bank* in *central bank* and a *bank* in *river bank*. The embeddings of *central bank* and *river bank* will be closed to each other even though their meanings are completely different. The keywords, which can compose of one or several words, can also be regarded as sentences. A better approach would be using sentence embeddings where instead of the individual words, the whole keyword is mapped to vectors of real numbers.

Sentence-BERT

In our study, we use a state-of-the-art sentence embedding technique called SBERT [28] to compute the vector representation of our keywords and our extracted sentences

[6] https://store.steampowered.com/app/428550/Momodora_Reverie_Under_The_Moonlight/

[7] https://store.steampowered.com/app/570940/DARK_SOULS_REMASTERED/

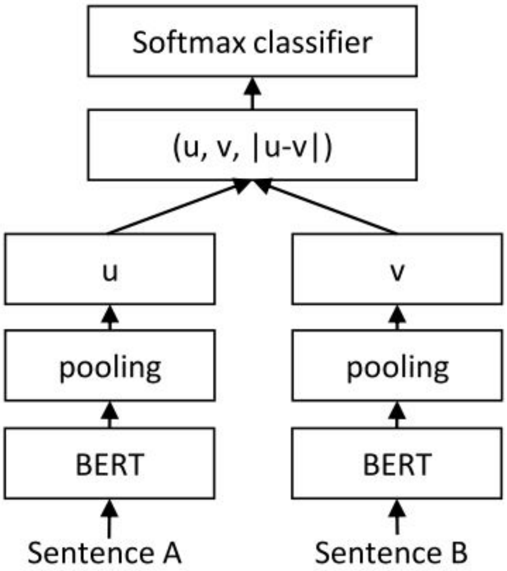


Figure 4.6: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure)

from the review. With the introduction of BERT, lots of tasks have tried to utilize this architecture, sentence embedding is no exception. A common method to address clustering and semantic search are to map each sentence to a vector space such that semantically similar sentences are close. Researchers have started to input individual sentences into BERT and to derive fixed-size sentence embeddings. The most commonly used approach is to average the BERT output layer (known as BERT embeddings) or by using the output of the first token (the `[CLS]` token). This common practice yields rather bad sentence embeddings, often worse than averaging GloVe embeddings. [28]

SBERT solves this problem by using a siamese network architecture that creates fixed-sized vectors for input sentences. Figure 4.6 shows the SBERT architecture. A siamese neural network (sometimes called a twin neural network) is an artificial neural network that uses the same weights while working in tandem on two different input vectors to compute comparable output vectors. The twin network helps fine-tune BERT by updating the weights such that the produced sentence embeddings are semantically meaningful and can be compared with cosine-similarity. SBERT is trained on the combination of the SNLI (Stanford Natural Language Inference) [4] and the Multi-Genre NLI [35] dataset. The SNLI is a collection of 570,000 sentence pairs annotated with the labels contradiction, entailment, and neutral. MultiNLI contains 430,000 sentence pairs and covers a range of genres of spoken and written text. Two sentences A and B are fed into the two BERT networks. Then a MEAN-pooling layer is applied, where the output of the `CLS`-token is used to compute the mean of all output vectors. We concatenate the sentence embeddings u and v with the element-wise difference $|u - v|$ and multiply it with the trainable weight $W_t \in \mathbb{R}^{3n \times k}$:

$$o = softmax(W_t(u, v, |u - v|))$$

where n is the dimension of the sentence embeddings and k the number of labels. We optimize cross-entropy loss. Semantically similar sentences can then be found using a similarity measure like cosine-similarity or Manhattan/Euclidean distance. These similarity measures can be performed extremely efficiently on modern hardware, allowing SBERT to be used for clustering.

4.2.3 Keywords clustering

(Explain what is agglomerative clustering, and HDBSCAN. Their advantages over other methods. Why choose agglomerative clustering over HDBSCAN in our study.)

For the task of clustering the keywords into groups, which we call aspects, clustering algorithms can be used. There are lots of approaches to clustering. An exhaustive list of different clustering algorithms can be found in [36]. But for our study, we only discuss two clustering algorithms that work well with our use cases for keywords clustering when doing aspect extraction and sentences clustering when doing text summarization. These are agglomerative clustering and HBDSCAN [6].

Agglomerative clustering

Agglomerative clustering is really a suite of algorithms all based on the same principle. The basic notion is that we start with each point in its own clusters and then, for each cluster, use some criterion to choose another cluster to merge with. Repeat until there is only one cluster left and we get a hierarchy, or binary tree, of clusters branching down to the last layer which has a leaf for each point in the dataset. The most basic version of this, single linkage, merges the clusters that are closest to each other, allowing the tree to be ranked by distance as to when clusters merged or split. To decide which cluster to merge, more complex variations use things like the mean distance between clusters, or

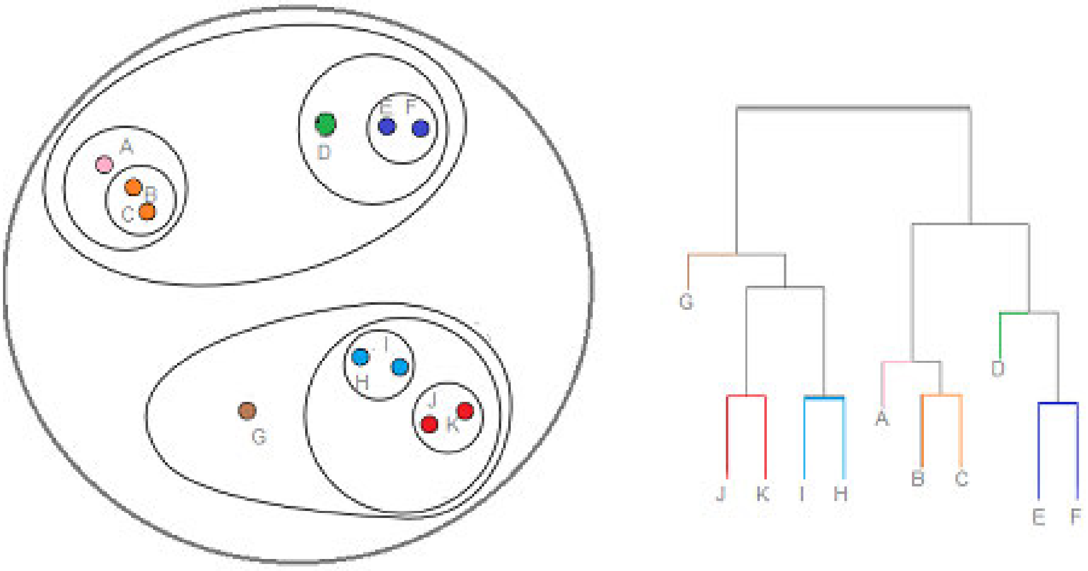


Figure 4.7: A dendrogram (right) representing nested clusters (left) [11]

distance between cluster centroids, etc. Once we have a cluster hierarchy, we can choose a level or cut (based on some criteria) and take the clusters at that level of the tree. An example of a cluster hierarchy can be seen in figure 4.7.

This approach has the advantage of allowing clusters to develop according to "the underlying manifold" rather than being assumed to be spherical. We may look at the dendrogram of clusters to get more information about how clusters break down and try to pick a natural cut, but this is similar to using the "elbow method" for finding the optimal k in K-Means: in theory, it is fine and simple, but when dealing with complex real-world data, it rarely works. The issue of noise polluting our clusters is there because we are partitioning rather than clustering the data. Contrary to the common belief that agglomerative clustering's performance is bad, it can be good if we get the right implementation. The sklearn[8] implementation is fairly slow, but fastcluster[9] provides high performance agglomerative clustering if needed.

HDBSCAN (Hierarchical density-based spatial clustering of applications with noise)

[8] https://scikit-learn.org/stable/

[9] https://pypi.org/project/fastcluster/

HDBSCAN is an algorithm recently developed by some of the same people who created the original DBSCAN [13], a popular density-based algorithm that assumes dense regions have clusters. DBSCAN's shortcomings include the fact that it does not function well with data with varying density and the distance parameter epsilon value must be chosen carefully. HDBSCAN solves these issues by allowing for varying density clusters. The algorithm begins similarly to DBSCAN: we transform the space according to data density, leaving points dense regions alone while moving points in sparse regions further away. A dendrogram is acquired by applying single linkage clustering on the transformed space. Rather than using an epsilon value as a cut level as DBSCAN does, the dendrogram is condensed by examining splits which results in a lower number of points splitting off as points "falling out of a cluster". As a result, the tree is smaller, with fewer clusters that "lose points". The tree can then be used to find the most stable and persistent clusters.

This method allows the tree to be cut at different heights, with the density clusters were chosen based on cluster stability. The immediate benefit is that we can have varying density clusters. The second advantage is that the epsilon parameter has been removed because we no longer need to pick a dendrogram cut. Instead, we have a new parameter `min_cluster_size` which is used to determine whether points are "falling out of a cluster" or splitting into two new clusters. This trades an unintuitive parameter for one that is not so hard to choose (what is the smallest cluster size we are willing to care about?) When implemented properly HDBSCAN can be very efficient.

Picking the clustering algorithm

After getting the sentence embeddings of each keyword, we normalize them to the unit length before passing them to the clustering algorithms. After closer inspection of the results from the two aforementioned algorithms, we choose agglomerative clustering for clustering keywords, as HDBSCAN identifies lots of meaningful and high-frequency keywords, which should have belonged to an aspect, as noise. With agglomerative clustering, the extracted aspect can contain a small number of keywords that should not be there. But this is acceptable as they tend to be outnumbered by the correct keywords, which will push them down the list when we order the most relevant keywords in an aspect by their frequencies. An aspect is named after its keyword with the highest frequency. One hundred aspects with the highest frequency are chosen for further analysis. Sentences that contain no keyword or keywords which have not been chosen are also removed.

4.3 Sentiment analysis

(Overview. Explain the challenges. Give examples.) The sentiment analysis step focuses on identifying the underlying sentiment that pervades each aspect. Since our aspects consist of keywords that belong to sentences, we perform sentence-level sentiment analysis.

We want to classify each sentence as *positive*, *negative*, or *neutral*. As there is no sentiment analysis dataset specific to our domain of video game reviews, we must either create our own labeled dataset for training or turn to unsupervised methods for sentiment classification.

Supervised approach

In order to create our own labeled dataset, several things should be taken into consideration. Each review on Steam is marked with either *Recommended* or *Not Recommended*. But each sentence in a review can express various sentiments, so labeling all sentences from a *Recommended* review as *positive* and all sentences from a *Not Recommended* review as *negative* is not a good solution. Furthermore, the number of sentences that express *neutral* sentiment is large and should not be ignored. It seems feasible at first to label a few thousand sentences ourselves for training, but this is not simply as a sentiment expressed in a sentence may be interpreted differently by different people, at different times and with a different mood, etc. And the number of sentences must also be really large, i.e. 10,000 upwards as video game reviews composes of many different genres and groups of players (hardcore, casual, etc.), reviewers (normal, professional, etc.). Another approach would be using crowdsourcing. This approach probably yields the best results for our sentiment analysis, but as described and explained in section 2.2, this process is costly, time-consuming, and also requires careful setup of data and questions. Using other datasets from similar domains like movie reviews, product reviews, app reviews, etc. is also possible. This approach proves to be really effective, but choosing the right dataset, finding or constructing the appropriate model, and fine-tuning it is non-trivial tasks.

Unsupervised approach

There are several approaches for unsupervised sentiment classification such as using a lexicon of sentiment words or pre-trained models. English has the advantage of being a popular language and therefore lots of libraries with pre-trained sentiment models. Here are some of them:

VADER [18]

VADER (Valence Aware Dictionary for Sentiment Reasoning) is a sentiment analysis tool that uses a lexicon-based approach (a lookup table of positive and negative words) with some simple heuristics (e.g. increasing the intensity of the sentiment if some words like *really*, *so*, or *a bit* a present). The benefit of this approach is that sentences containing negated positive words (e.g. *not happy*, *not good*) will still receive a negative sentence sentiment thanks to the heuristics to flip the sentiment of the word following a negation. Some simpler sentiment analysis tools will just take the average of the sentiments of the words and would miss subtle details like this. The drawback of this approach is that words which the sentiment analysis tool has not seen before will neither be classified as *positive* nor *negative* (e.g. typos). The model is trained using social media text and as a result, yields sub-optimal results when working with our video game reviews.

TextBlob[10]

Textblob's sentiment analysis works in a similar way to VADER, which is using a lexicon-based classifier, but the advantage is that it includes subjectivity analysis too (how factual/opinionated a piece of text is). However, it does not contain the heuristics that VADER has, and so it will not intensify or negate a sentence's sentiment.

Flair[11] Flair's sentiment classifier is based on a character-level RNN which takes sequences of letters and words into account when predicting. The network has learned to take negations into account as well as intensifiers. But probably one of its biggest advantages is that it can predict sentiment for words that it has never seen before too (such as typos). There is also a transformer-based model, which is slower, but slightly more accurate. The models are trained over a combined corpus of sentiment datasets, including movie reviews and Amazon product reviews[12]. This makes the models more applicable in a much wider range of domains, even ones that have not been trained on. Upon testing with our video game reviews, the transformer-based model performs better than the RNN-based one when predicting positive and negative sentiments. However, the transformer-based model fails to predict neutral sentences and always gives a very

[10] https://textblob.readthedocs.io/en/dev/
[11] https://github.com/flairNLP/flair
[12] https://github.com/flairNLP/flair/releases

polarized score, which leads to its predicting neutral sentences as either positive or negative. The RNN works slightly less accurately with positive and negative sentiments, but for neutral sentences, the model successfully delivers a neutral score and thus will be used for the text of our review.

Using Flair's RNN-based sentiment model, we classify all the remaining sentences from the previous step as *positive*, *negative*, or *neutral.*

4.4 Text summarization

This section explains how we create customized, aspect-based summaries of video game reviews. The purpose of our summary is to capture the main opinions about each aspect of the reviews. Traditional text summarization, on the other hand, selects a subset or rewrites some of the original sentences to capture the main points. Existing techniques of aspect-based summarization generate summaries based solely on the features of the text, with little consideration for the reader. This lack of customization is a drawback as different people often desire different content for their summaries. Each person has a different perspective on the same text and highlights the need for customization of summaries. In this study, we achieve customization by ensuring that summaries cover the aspects the users are interested in.

4.4.1 Sentences embedding

Our approach for summarization is based on clustering algorithms, therefore the sentences need to be transformed into real-value vectors first. For this task, we again use SBERT, which has been mentioned and discussed in detail in the section 4.2.2.

4.4.2 Sentence clustering

Our method is extractive, i.e. it selects a subset of sentences from all the available sentences of a game's aspect. We take as input a set of sentences, each of which has been annotated with not only an aspect but also other information such as the time it was written, user's recommendation, sentiment polarity using the methods described in the previous sections. All of this information, depending on the user's interest, can act as filters for the set of extracted sentences. In order to avoid redundancy and cover as many different opinions as possible, we go through each aspect and try to find the most salient sentences. This also helps alleviate the computational strain as performing clustering multiple times with a smaller set of sentences is faster than clustering once with all the available sentences. With the most important sentences for each aspect identified, we can create a personalized summary with all the aspects that the user is interested in. As a departure from previous studies, we explicitly select the sentences that will form the basis for summarization, in the following steps.

Sentences merging: To avoid selecting redundant sentences, we apply HDBSCAN to merge similar sentences into clusters. For each sentence cluster, we define its representative sentence which is the sentence closest to its centroid.

Sentence ranking: We assume that larger clusters contain sentences that are popular among reviews and, therefore, should have higher priority to be included in the set of most salient opinions. We use the representative sentences of the top-k largest clusters, as selected sentences. In the very rare cases where we are trying to get k clusters, but HDBSCAN delivers less than k clusters, we must use agglomerative clustering with a distance threshold of 0.6, decreasing by 0.1 until the clusters are fine-grained enough and the number of required clusters k is met. With HDBSCAN, the strength of having few parameters to set is also one of its weaknesses. If the result is not satisfactory, there are a few parameter that we can try to tune like `min_cluster_size`, `min_samples`, `cluster_selection_epsilon` or `alpha`. Except for the 2 intuitive parameters `min_cluster_size`, `min_samples` which we usually pick at the beginning, the remaining 2 are quite unintuitive. That is why we need agglomerative clustering with its simple parameter distance threshold.

Sentence filtering: As each sentence is annotated with not only an aspect but also other information such as the time it was written, user's recommendation, sentiment polarity using the methods described in the previous sections. All of this information, depending on the user's interest, can act as filters for the set of extracted sentences. This comes in handy when we come to our data visualization as users can select and filter sentences as they see fit on the dashboard.

Summarization: At summarization time, we pick out the k most salient sentences from each aspect and use them as summarization. The users can interact with the dashboard and choose how many aspects, how many sentences for each aspect, etc. they want to see. The summarization is really efficient and can be computed on the fly when there is a user's interaction with the dashboard.

4.5 Data visualization

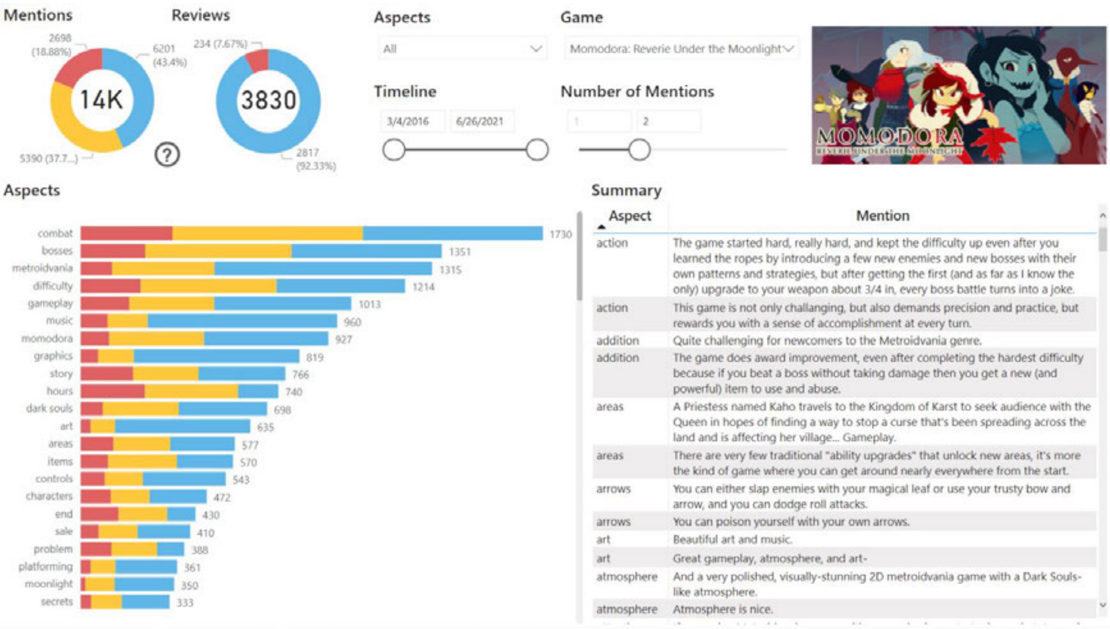
The final component of our system is data visualization. After obtaining all the necessary data and building appropriate models from the previous steps, we need to find a convenient way for the users to explore and interpret the results. In this study, we build an interactive dashboard that lets users explore the aspect-based sentiment analysis and text summarization of a given game. The users in this case can be potential customers who are considering buying that game and want to see what kind of mechanics, features, etc. it offers. Another group of users are developers who want to know how their game is doing, what their customers like or dislike about it. With this knowledge, they can quickly pinpoint problems or things that their customers actually care about in their game and learn valuable lessons for their next productions.

Existing summarization systems often fall short on two aspects. First, existing techniques generate static summaries which cannot be tailored to specific user needs. Second, most existing systems generate extractive summaries which select only certain salient aspects from the summaries. Hence, they do not completely depict the overall opinions of the reviews. Our system tries to overcome these shortcomings by allowing the user to interact with the dashboard and generate tailored summaries on the fly.

For the choice of dashboard development, we employ Power BI for ease of use and quick development. There are much more powerful Python tools suitable for this task such as Plotly Dash[13], Streamlit[14], Voilà[15], Panel[16], etc. All offer really good customization when creating dashboards, but also require a lot more time and effort for development, deployment, and maintenance. A Power BI dashboard can act as a prototype for further development of the visualization in another tool.

Our dashboard consists of two pages, which are Overview and Mention Analysis. We will go into the details about all the components on each page and explain how they can be used and work together.

[13] https://plotly.com/dash/
[14] https://streamlit.io/
[15] https://github.com/voila-dashboards/voila
[16] https://panel.holoviz.org/



Aspect	Mention
action	The game started hard, really hard, and kept the difficulty up even after you learned the ropes by introducing a few new enemies and new bosses with their own patterns and strategies, but after getting the first (and as far as I know the only) upgrade to your weapon about 3/4 in, every boss battle turns into a joke.
action	This game is not only challanging, but also demands precision and practice, but rewards you with a sense of accomplishment at every turn.
addition	Quite challenging for newcomers to the Metroidvania genre.
addition	The game does award improvement, even after completing the hardest difficulty because if you beat a boss without taking damage then you get a new (and powerful) item to use and abuse.
areas	A Priestess named Kaho travels to the Kingdom of Karst to seek audience with the Queen in hopes of finding a way to stop a curse that's been spreading across the land and is affecting her village... Gameplay.
areas	There are very few traditional "ability upgrades" that unlock new areas, it's more the kind of game where you can get around nearly everywhere from the start.
arrows	You can either slap enemies with your magical leaf or use your trusty bow and arrow, and you can dodge roll attacks.
arrows	You can poison yourself with your own arrows.
art	Beautiful art and music.
art	Great gameplay, atmosphere, and art-
atmosphere	And a very polished, visually-stunning 2D metroidvania game with a Dark Souls-like atmosphere.
atmosphere	Atmosphere is nice.

Figure 4.8: Overview page of the dashboard

Overview page

Figure 4.8 shows the Overview page of our dashboard. This is the page the user will come into contact with first upon using the dashboard. It shows the most important information about a game such as the total number of sentences (we call them *mentions* in this case), the total number of reviews, the sentiment among different aspects, and the summary which composes of the most important mentions for each aspect. Blue is for positive sentiment, red is for negative, and yellow is for neutral. Here are the details of each component:

- **Game selector**: (Figure 4.9) A single-select drop-down for choosing a game to analyze.
- **Aspects selector**: (Figure 4.10) A multi-select drop-down for choosing the aspects in which we are interested.
- **Timeline selector**: (Figure 4.11) A time slider for choosing the time range of interest.

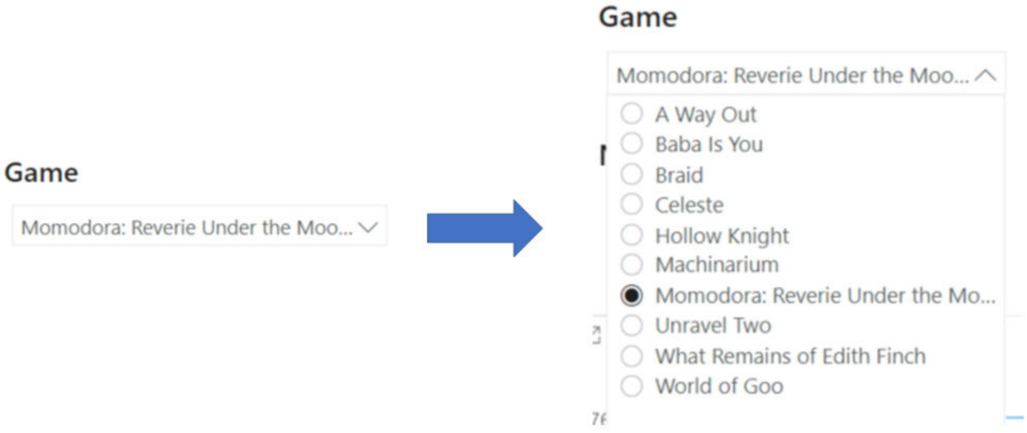

Game
Momodora: Reverie Under the Moo...
Game
Momodora: Reverie Under the Moo...
○ A Way Out
○ Baba Is You
○ Braid
○ Celeste
○ Hollow Knight
○ Machinarium
● Momodora: Reverie Under the Mo...
○ Unravel Two
○ What Remains of Edith Finch
○ World of Goo

Figure 4.9: Game selector

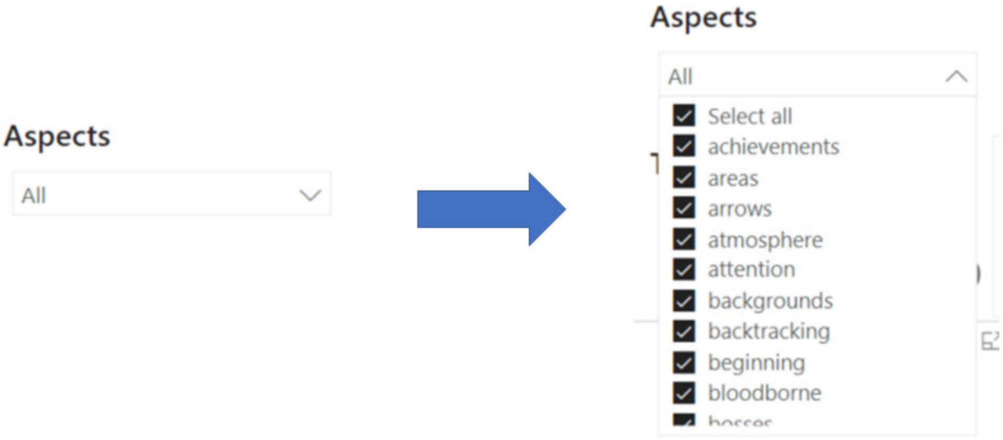


Figure 4.10: Aspects selector

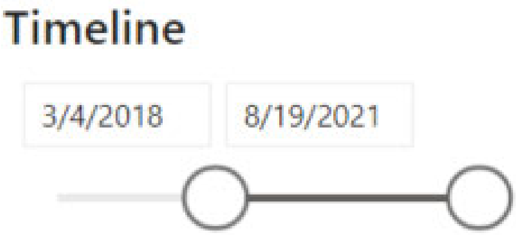
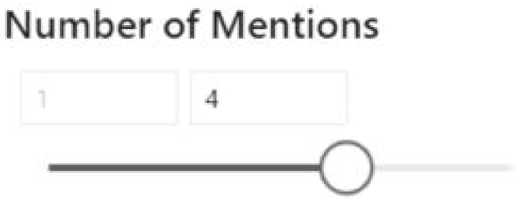


Figure 4.11: Timeline selector


Number of Mentions

1

4

Figure 4.12: Number of mentions selector

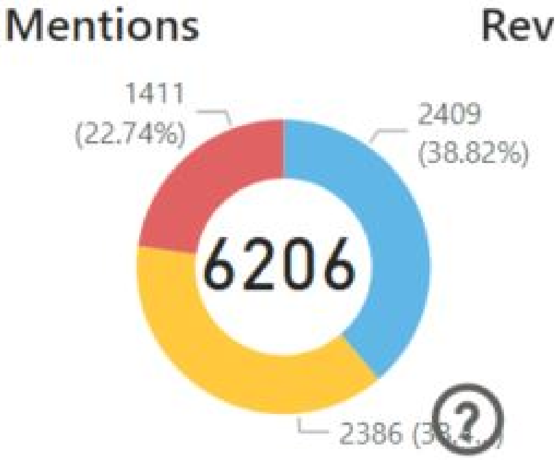


Figure 4.13: Mentions overview

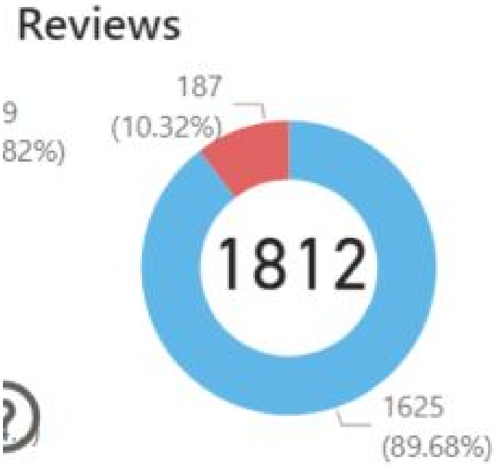
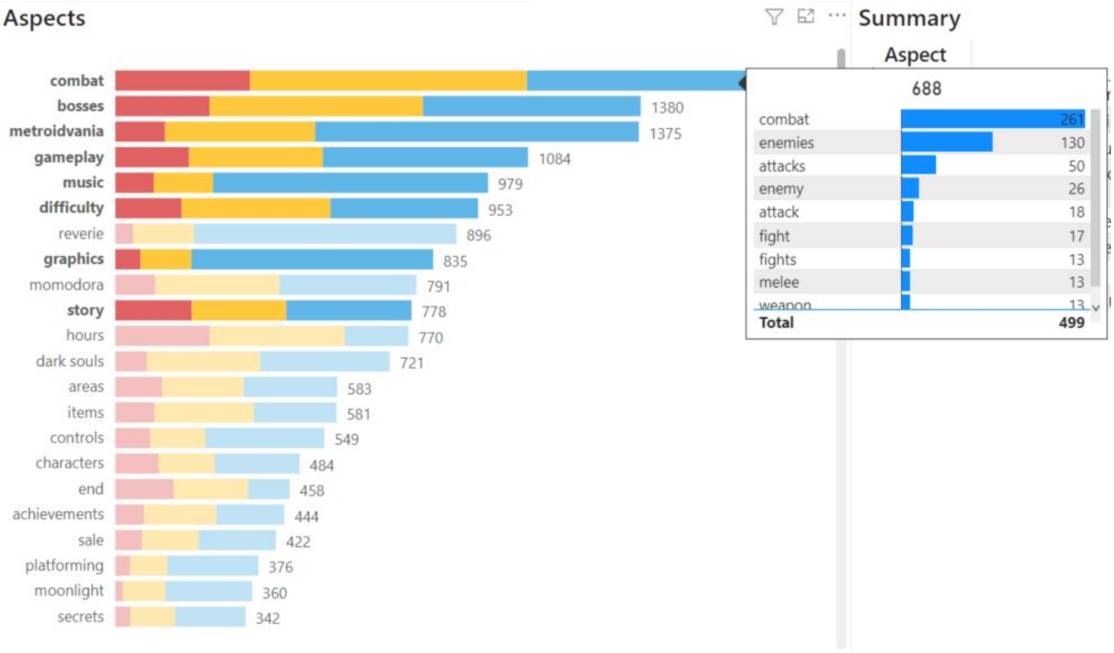
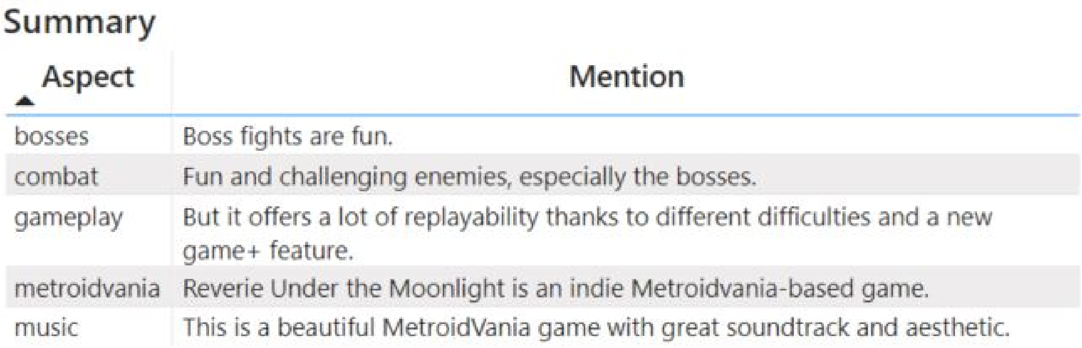


Figure 4.14: Reviews overview



688	
combat	261
enemies	130
attacks	50
enemy	26
attack	18
fight	17
fights	13
melee	13
weapon	13
Total	**499**

Figure 4.15: Aspects sentiment chart



Summary

Aspect	Mention
bosses	Boss fights are fun.
combat	Fun and challenging enemies, especially the bosses.
gameplay	But it offers a lot of replayability thanks to different difficulties and a new game+ feature.
metroidvania	Reverie Under the Moonlight is an indie Metroidvania-based game.
music	This is a beautiful MetroidVania game with great soundtrack and aesthetic.

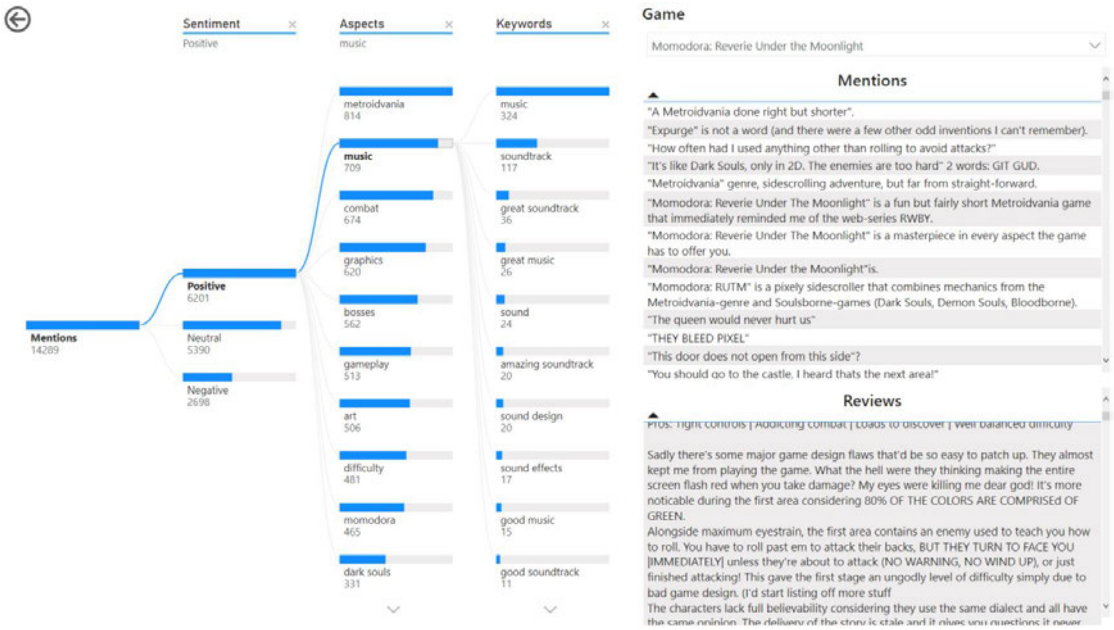
Figure 4.16: Summary

- **Number of mentions selector**: (Figure 4.12) A number slider for choosing how many mentions for each aspect we want to see. It goes from a minimum of 1 to a maximum of 5.
- **Mentions overview**: (Figure 4.13) A donut chart that shows the proportion of each sentiment and the total number of mentions in the middle. The question mark can be clicked to open the Mention Analysis page which enables us to dive deeper into the details of those mentions.
- **Reviews overview**: (Figure 4.14) A donut chart that shows the proportion of *Recommended* and *Not recommended* reviews with the total number of reviews in the middle.
- **Aspects sentiment chart**: (Figure 4.15) A horizontal bar chart that shows the proportion of each sentiment for each aspect. The total number of mentions of each aspect is also displayed at the end of each bar chart. For a given game, we take 100 aspects in total, but only the top 22 are visible in the chart. The rest and less important aspects can be found by scrolling down the bar chart. Upon hovering on a bar, the top 8 most frequent keywords of that aspect are also shown. The chart can also be used for choosing the aspects that we care about instead of manually using the aspects selector. This helps us reduce the number of mentions shown in the summary.
- **Summary**: (Figure 4.16) A table where each row is a mention with the aspect it contains. This will act as a summary as we use the aspects selector or the aspects sentiment chart to select the aspects of interest.

Mention Analysis page

Figure 4.17 shows the Mention Analysis page of our dashboard. The page can be accessed by clicking the question mark symbol in the *mentions overview.* It helps us inspect the mentions by sentiment, aspects, or keywords. Here are the components of this page:

- **Mentions decomposition tree**: (Figure 4.18) The tree starts at the root with all the mentions. From here, we can break down the number of mentions by 3 factors which are *Sentiment*, *Aspects*, and *Keywords.* It automatically aggregates the number of mentions and enables drilling down into these 3 factors in any order. Moreover, we can let the tree automatically pick the next factor to drill down into based on certain criteria such as the highest or lowest number of mentions.



Game

Momodora: Reverie Under the Moonlight

Mentions

"A Metroidvania done right but shorter".

"Expurge" is not a word (and there were a few other odd inventions I can't remember).

"How often had I used anything other than rolling to avoid attacks?"

"It's like Dark Souls, only in 2D. The enemies are too hard" 2 words: GIT GUD.

"Metroidvania" genre, sidescrolling adventure, but far from straight-forward.

"Momodora: Reverie Under The Moonlight" is a fun but fairly short Metroidvania game that immediately reminded me of the web-series RWBY.

"Momodora: Reverie Under The Moonlight" is a masterpiece in every aspect the game has to offer you.

"Momodora: Reverie Under the Moonlight"is.

"Momodora: RUTM" is a pixely sidescroller that combines mechanics from the Metroidvania-genre and Soulsborne-games (Dark Souls, Demon Souls, Bloodborne).

"The queen would never hurt us"

"THEY BLEED PIXEL"

"This door does not open from this side"?

"You should go to the castle. I heard thats the next area!"

Reviews

Pros: Tight controls | Addicting combat | Loads to discover | Well balanced difficulty

Sadly there's some major game design flaws that'd be so easy to patch up. They almost kept me from playing the game. What the hell were they thinking making the entire screen flash red when you take damage? My eyes were killing me dear god! It's more noticable during the first area considering 80% OF THE COLORS ARE COMPRISEd OF GREEN.
Alongside maximum eyestrain, the first area contains an enemy used to teach you how to roll. You have to roll past em to attack their backs, BUT THEY TURN TO FACE YOU [IMMEDIATELY] unless they're about to attack (NO WARNING, NO WIND UP), or just finished attacking! This gave the first stage an ungodly level of difficulty simply due to bad game design. (I'd start listing off more stuff
The characters lack full believability considering they use the same dialect and all have the same opinion. The delivery of the story is stale and it gives you questions it never

Figure 4.17: Overview page of the dashboard

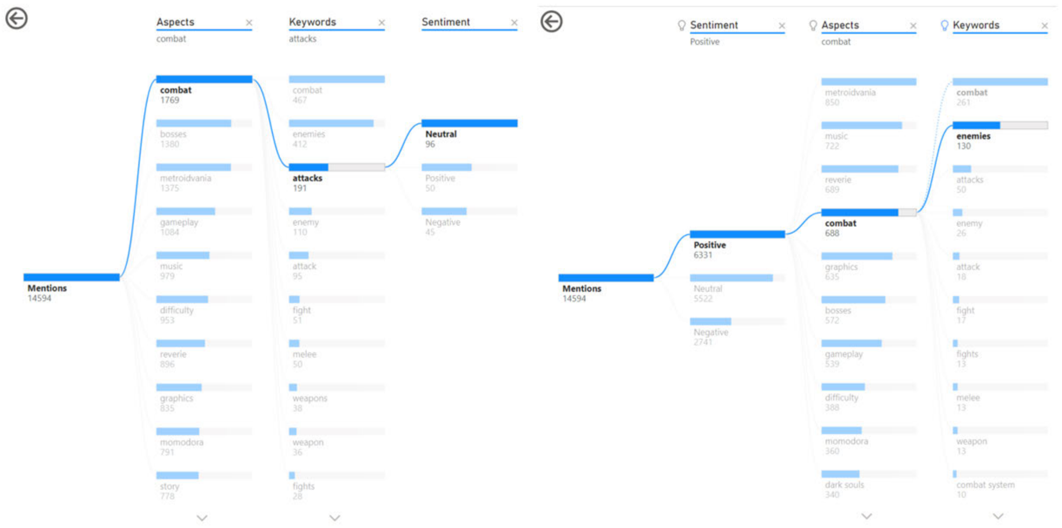
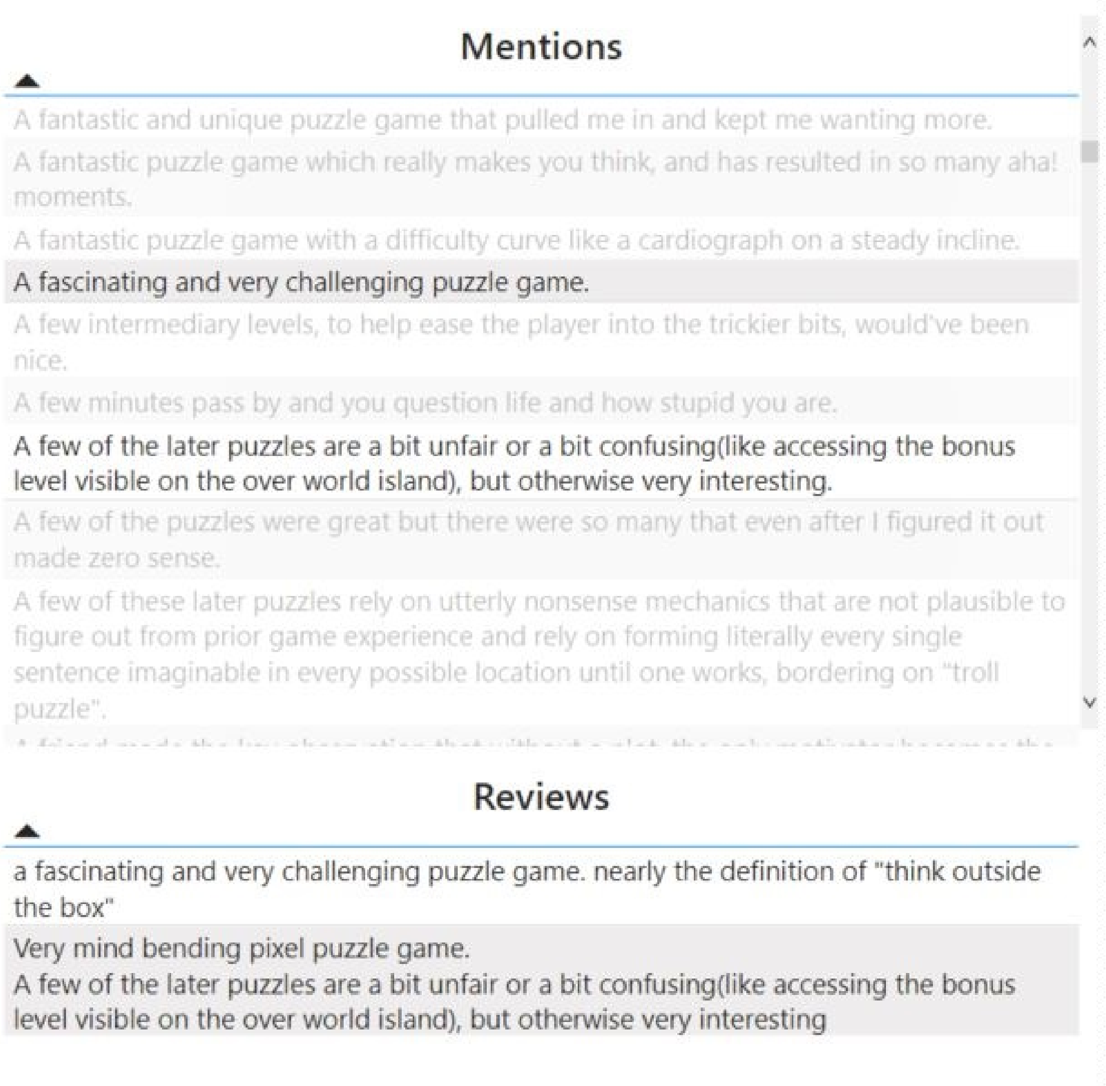


Figure 4.18: Mentions decomposition tree



Mentions

▲

A fantastic and unique puzzle game that pulled me in and kept me wanting more.

A fantastic puzzle game which really makes you think, and has resulted in so many aha! moments.

A fantastic puzzle game with a difficulty curve like a cardiograph on a steady incline.

A fascinating and very challenging puzzle game.

A few intermediary levels, to help ease the player into the trickier bits, would've been nice.

A few minutes pass by and you question life and how stupid you are.

A few of the later puzzles are a bit unfair or a bit confusing(like accessing the bonus level visible on the over world island), but otherwise very interesting.

A few of the puzzles were great but there were so many that even after I figured it out made zero sense.

A few of these later puzzles rely on utterly nonsense mechanics that are not plausible to figure out from prior game experience and rely on forming literally every single sentence imaginable in every possible location until one works, bordering on "troll puzzle".

Reviews

▲

a fascinating and very challenging puzzle game. nearly the definition of "think outside the box"

Very mind bending pixel puzzle game.
A few of the later puzzles are a bit unfair or a bit confusing(like accessing the bonus level visible on the over world island), but otherwise very interesting

Figure 4.19: Mentions and reviews list

- **Game selector**: (Figure 4.9) Similar to the game selector of the Overview page. These two selectors are also connected.
- **Mentions list**: (Figure 4.19) A list of mentions. By clicking a node in the decomposition tree, the list is filtered and shows us all the mentions in that node.
- **Reviews list**: (Figure 4.19) A list of reviews. By clicking one or more mentions in the Mentions list, one or more reviews, from which the mentions come, are shown.

5 Experiments and evaluation

(We apply with our pipelines to n different more games, which have a varied number of reviews, players' sentiment. What is our bottleneck? (the aspect extraction step which requires the BERT model))

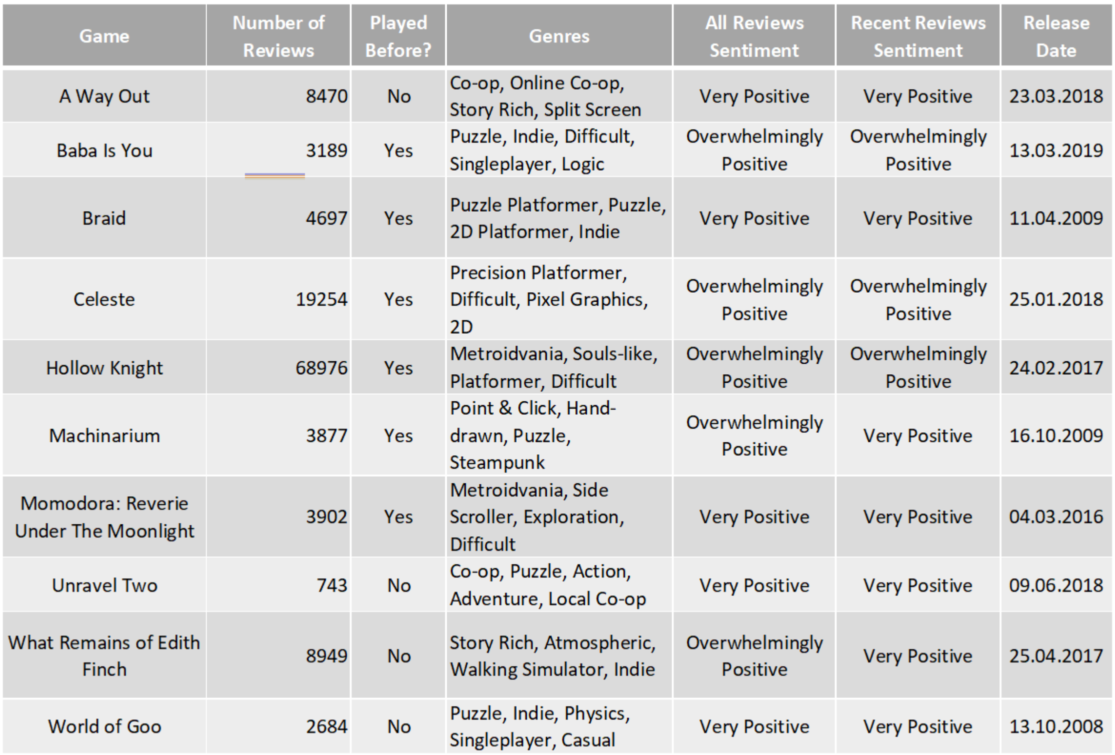
(Intrinsic vs extrinsic evaluation?) (Compare our system with other related works.) (Evaluate the results)

5.1 Dataset

We collect all the reviews written in English of 12 different games on Steam and put them in the system. The dataset includes many different types of games from different genres to ensure as much variety as possible. Moreover, the games are also chosen based on the fact that we have played most of them and can have a rough idea about the games like their features, strengths, weaknesses, etc. Some games which have never been played before are also included to avoid confirmation bias while evaluating the effectiveness of the system as we tend to search for, interpret, favor results in a way that confirms or supports our prior knowledge or assumptions about the ones that have been played. The games also vary in the number of reviews. Some with only hundreds, others with tens of thousands. For a complete list of all the games used in our study and information about them, see figure 5.1.

5.2 Aspect extraction

Using a transformer model for POS tagging during keywords extraction has proven to be the bottleneck of our system as it is very computationally expensive to perform this task. Our initial approach for reducing computational time is to set a limit of a maximum of



Game	Number of Reviews	Played Before?	Genres	All Reviews Sentiment	Recent Reviews Sentiment	Release Date
A Way Out	8470	No	Co-op, Online Co-op, Story Rich, Split Screen	Very Positive	Very Positive	23.03.2018
Baba Is You	3189	Yes	Puzzle, Indie, Difficult, Singleplayer, Logic	Overwhelmingly Positive	Overwhelmingly Positive	13.03.2019
Braid	4697	Yes	Puzzle Platformer, Puzzle, 2D Platformer, Indie	Very Positive	Very Positive	11.04.2009
Celeste	19254	Yes	Precision Platformer, Difficult, Pixel Graphics, 2D	Overwhelmingly Positive	Overwhelmingly Positive	25.01.2018
Hollow Knight	68976	Yes	Metroidvania, Souls-like, Platformer, Difficult	Overwhelmingly Positive	Overwhelmingly Positive	24.02.2017
Machinarium	3877	Yes	Point & Click, Hand-drawn, Puzzle, Steampunk	Overwhelmingly Positive	Very Positive	16.10.2009
Momodora: Reverie Under The Moonlight	3902	Yes	Metroidvania, Side Scroller, Exploration, Difficult	Very Positive	Very Positive	04.03.2016
Unravel Two	743	No	Co-op, Puzzle, Action, Adventure, Local Co-op	Very Positive	Very Positive	09.06.2018
What Remains of Edith Finch	8949	No	Story Rich, Atmospheric, Walking Simulator, Indie	Overwhelmingly Positive	Very Positive	25.04.2017
World of Goo	2684	No	Puzzle, Indie, Physics, Singleplayer, Casual	Very Positive	Very Positive	13.10.2008

Figure 5.1: List of games used in our study and additional information about them

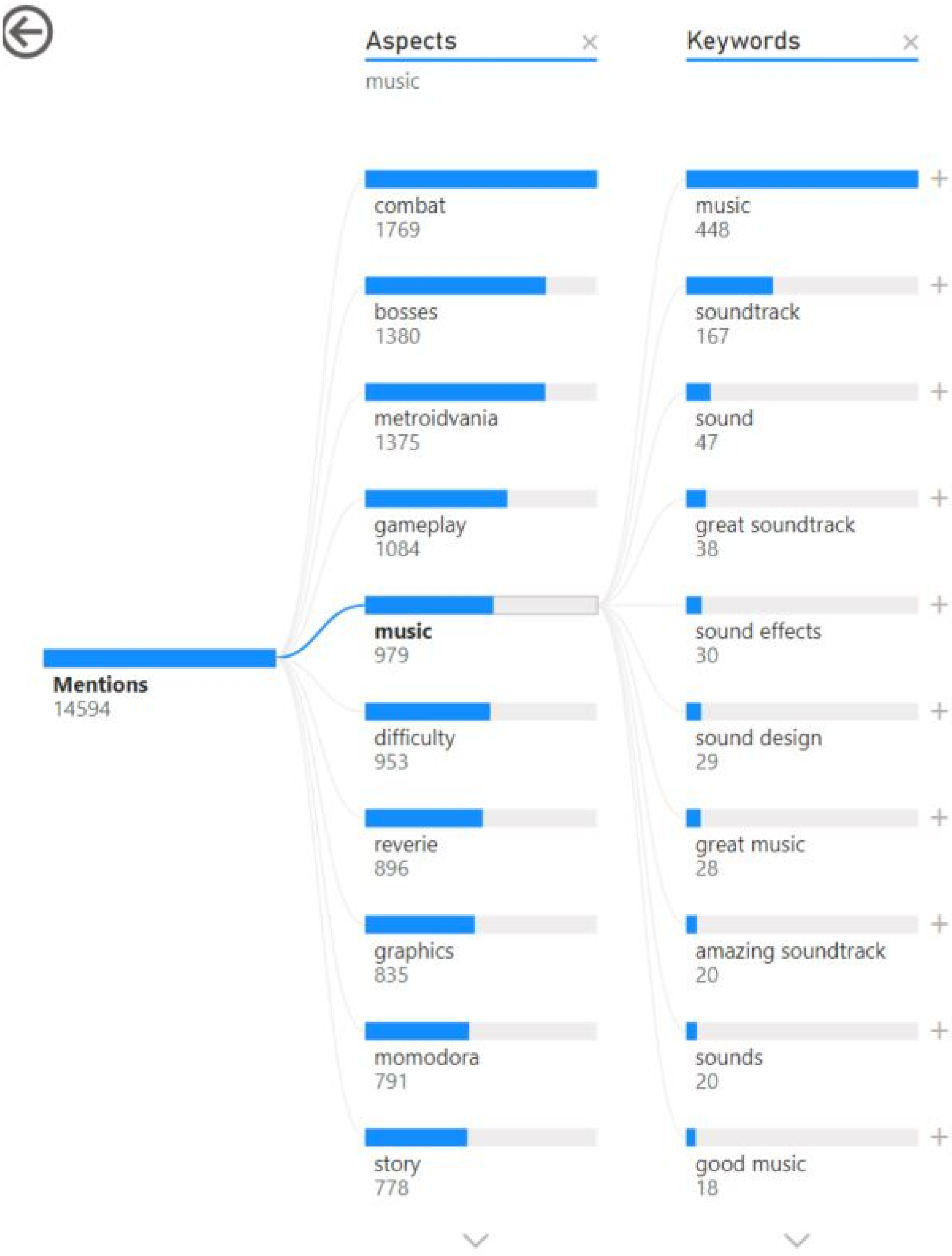


Figure 5.2: Using the mentions decomposition tree to check the quality of our aspect extraction

5000 newest reviews for each game. But this is ineffective as games with lots of reviews can accumulate 5000 reviews in just a couple of days and taking only these reviews into consideration does not reflect the overall characteristics of those games very well. The majority of reviews posted are short and often contribute nothing to our understandings of a game (e.g. *"Nice."*, *"Good!"*, *"Bad game!"*, *"LoL."*, *"I love it."*, *"The game sucks!"* etc.). And thus during aspect extraction, not a lot of keywords or aspects can be extracted from games with lots of reviews in comparison to games with fewer reviews, but all or most of which have been analyzed by the system. So for our experiments, we do not limit the number of reviews anymore and take all the available reviews for each game into consideration. After testing different distance thresholds for the agglomerative clustering to group similar keywords together, we use 0.6.

As the system is unsupervised, we do not have the labeled test set to compute metrics such as accuracy, precision, or recall for this task. We try to go through the top 10 extracted aspects of each game and carefully inspect the top 10 keywords of each of those aspects to see if the aspects and the keywords that make them up are coherent. With the help of the mentions decomposition tree in the dashboard, this task can be performed easily by drilling down the *Aspects* and *Keywords* factors. Figure 5.2 shows the tree in action. It has been found that the name of a game or keywords that belong to that name is often one of the biggest clusters. We can easily set them as stop words, but sometimes some keywords in the name of the game are actually important features or mechanics that should not be removed. Our system allows us to control and filter the extracted aspects according to our preferences, if an unwanted aspect appears, you just have to remove it from the selection. Besides the name of the game, there are also a few keywords that should have been classified as stop words but are not such as *hours*, *plenty*, *kind*, etc. This is not a big problem as the frequency of the most important aspects heavily outweighs these tiny amounts of stop words.

The aspects extracted in the games that we have played and already knew about are great as they capture the main features of those games. For example, with *Momodora: Reverie Under the Moonlight* we can quickly see that lots of players talk about *combat*, *bosses*, *music*, *metroidvania*, *dark souls*, etc. as the game is a 2D metroidvania side scroller that takes lots of inspiration from the series *Dark Souls.* The boss fights and soundtracks are also some of the highlights of the game. For *Baba is You*, the most important keywords are *puzzle*, *baba*, *level*, *brain*, *solution*, *rules*", etc. as it is a Sokoban puzzle game where the rules and mechanics of the game can be changed by forming different sentences by pushing the word blocks. The keywords are sometimes grouped together incorrectly when a keyword has two completely different meanings. For example, the game *Hollow Knight* has a world where everyone is insects or *bugs*. But *bugs* can also be errors, flaws, or faults in the game. Even though SBERT introduces context, the keywords can come in a single word such as *bugs*, *bank*, *souls*, etc. and there is no context for SBERT to distinguish these two like in *central bank* and *river bank*.

For games that we have never played before, we can infer from the extracted keywords what their main features are. For *What Remains of Edith Finch*, the aspect *story* greatly outweighs others so we can infer that the game places a heavy focus on telling an intriguing story. Other aspects such as *art*, *walking simulator* gives us an idea about what makes the game special. With *Unravel Two*, the most important aspects are *friend*, *multiplayer*, *puzzles*, *co op*. This quickly helps us understand that the game is a co-op puzzle game, which is great for playing with friends.

5.3 Sentiment analysis

For the task of sentiment analysis, using the Flair RNN model provides us much better performance not only in terms of accuracy but also speed as explained in section 4.3. Another advantage of using these pre-trained model is that the data preprocessing steps is built in. So there is no need for complicated preprocessing steps.

The RNN-model gives predictions ranging from -1 to 1, with -1 being the most negative and 1 being the most positive. Upon trying out the model on different sets of sentences, we classify anything above 0.9 as *positive*, anything below -0.9 as *negative*. The rest is classified as *neutral*.

We do not have a dataset of sentences labeled with three sentiments *neutral*, *positive*, and *negative* available in the domain of video games. Creating enough labeled data with great quality for validation is a non-trivial task as labeling the sentences ourselves would introduce a lot of error and bias. Using crowdsourcing platforms and labeling a big and diverse enough amount of data is time-consuming and costly. In this study, using the dashboard created, we randomly go through some sentences and check if the predicted sentiments are correct. We can see that the model successfully detected the sentiments in most of the sentences correctly. In cases where there is sarcasm, the model can sometimes understand it. There are a few edge cases where the model fails to understand the true meaning of a word because the context is missing from the sentences. An example would be *"Bugs are everywhere."* talking about the game *Hollow Knight* where every character is a sort of insect. The model would mistake *bugs* as errors, flaws in the game and classify this sentence as negative. There are cases where the sentences are not correctly preprocessed and appear with bad structure. In such cases, the model can still pick out the parts that yield sentiment scores and classify the sentiment correctly.

5.4 Text summarization

Our system can theoretically deliver a personalized summary on the fly based on the user's preference when using the dashboard. Power BI does allow us to integrate Python code into the visuals, but the problem is things such as importing libraries, loading the clustering models, declaring environment variables, or converting the sentences from JSON format to vectors, etc. cannot be precomputed and saved somewhere when loading the dashboard. These quite computationally heavy tasks are repeated every time our data changes which greatly harms the user's experience. In this study, we are using Power BI as a prototype tool for ease of use and speed of development. Other more robust tools for the tasks such as Plotly Dash, Streamlit, etc. can easily solve this problem. As we want to preserve a smooth user experience, we precompute and save the summarization, which is the 5 most salient sentences, for each aspect of a game in the database and just loads them when needed instead of computing them on the fly with personalized filters. This works fine at first glance but may deliver no summary at all if the salient sentences are not present in the filtered set of sentences.

For agglomerative clustering, we set the distance threshold to 0.5, instead of 0.6 during aspect extraction as sentences contain more characters and thus more complexity than short keywords. The distance threshold must therefore be stricter to correctly cluster really similar sentences.

For text summarization, we usually use the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [34] score to compare the summarized text generated by our model with gold standard summaries. But the availability of such dataset is really rare. Trying to construct one is not the best idea as there, are just so many ways to create a good summary for a piece of text. The ROUGE metrics only captures how similar two pieces of text to each other on a word basis. But sentences can be completely rewritten and still convey the full meaning of the original text. Hence, the best way to compare different approaches is to create our own evaluation set or ask human annotators to rate the summaries produced by different algorithms in terms of coherence, accuracy of the summary, etc. For our summaries, we try to maximize coverage by selecting sentences from different aspects and minimize redundancy by clustering similar sentences. The summary for each aspect is good. But sometimes a sentence can contain two keywords, each with different aspects. The unwanted aspect can be mixed in with the correct one in a summary, e.g. *"Arts and music are great."*. Sometimes sentences that have not been preprocessed correctly can appear multiple times and become summary sentences. Such sentences often come from some sort of review form that has really complicated structure and trying to preprocess them is a challenging task. Otherwise, the summaries are of good quality.

6 Conclusion

6.1 Summary

The primary target of this work is to construct a machine learning system for aspect-based sentiment analysis and text summarization of video game reviews found on Steam.

This project utilizes simple, yet effective unsupervised methods for performing aspect-based sentiment analysis and text summarization of the video game reviews. The 4 main components of the system are:

- **Aspect extraction**: We use a heuristic-based approach that utilizes the part-of-speech (POS) tagging capability of a transformer-based model in SpaCy and certain rules to extract noun chunks. We call these noun chunks keywords. We then compute the sentence embeddings of each extracted keyword using SBERT and apply agglomerative clustering to cluster similar ones together into groups, which are called aspects. Even though lots of computational time can be saved by using the POS tagging of a simpler model than a transformer-based one, the extracted aspects suffer a lot in terms of accuracy. The majority of the extracted aspects are meaningful as they contain groups of keywords that are coherent and describe or represent the main features, interests, or concerns of the players. A small number of keywords are actually stopwords that can be filtered out easily. Sentences, in which the keywords are found, are also extracted.

- **Sentiment analysis**: An RNN-based pre-trained sentiment classification model is used to classify the sentences extracted from the previous step as *positive*, *negative*, or *neutral*. The model has no problem in terms of speed. In terms of accuracy, the model manages to detect the sentiments of the majority of sentences correctly, even ones where the sentiments are implicit or the reviewers use sarcasm. Misclassified sentences are really rare.

- **Text summarization**: For text representation, we use SBERT to compute the embedding of each sentence. HDBSCAN is used first to perform clustering on these sentences. We then find the sentence lying closest to the center of each cluster and call it an opinion. We rank the opinions by the size of their clusters. We assume that the bigger the cluster, the more popular an opinion is. And thus for each extracted aspect from the previous step, we have a list of n most important opinions, which will act as a summary for that aspect. If HDBSCAN fails to find enough n clusters, we will use agglomerative clustering with decreasing distance threshold until we have enough n clusters. This approach works great as it avoids redundancy by clustering similar sentences together and maximizes coverage by picking out the most important opinions of each aspect. The performance also improves as we perform clustering on smaller sets of sentences instead of the whole dataset, which enables us to generate a summary on the fly as we click and filter the sentences by different criteria when interacting with the visualization.

- **Data visualization**: We put together an interactive dashboard that helps us visualize the results and perform further analysis. The dashboard is implemented as a prototype in Power BI, which offers a variety of useful premade visuals. The dashboard composes of 2 pages: an overview page for us to see the overall sentiment of each aspect and generate a summary according to our preferences, a mentions analysis page for us to dive deeper into the extracted keywords and their respective sentences (or mentions) and reviews. The summary can theoretically be generated on the fly as we interact with the dashboard and filter the set of sentences by aspects, sentiments, time, etc. But due to the constraints of Power BI, this is not possible and will damage the performance and user experience of the dashboard. So we opt for another solution, which is precomputing a summary of the 5 most important sentences for each aspect of each game. This helps us quickly find the most important sentences as we interact with the dashboard, but we cannot generate a summary of the filtered set of sentences anymore.

6.2 Future work

The results of this study point to a number of open problems to generate good opinion summarization of user reviews. The tools and techniques proposed are just an initial step towards building a complete system that allows customers and developers to quickly analyze reviews using aspect-based sentiment analysis, text summarization, and data visualization. Here are a number of research directions:

- **Genres, users analysis**: Other information about a game such as genres or the user's profile of the reviewers can also be extracted. The dynamics of different genres and user groups between different games can be found by carefully analyzing these data.

- **Data streaming**: The Steam platform generates an unfathomable amount of reviews every minute of every day, and it continues to multiply at a staggering rate. That is why there is a need for shifting from batch processing to stream processing, which means the continuous flow of data generated in real-time can be processed, stored, analyzed, and acted upon.

- **Avoiding data drift**: Data drift is a change in the distribution of a baseline data set on which the model was trained and the current real-time production data. As our system is unsupervised, this problem is greatly mitigated. But as the opinions of players about a game change, so as the underlying distribution of the data. We need to find a way to update the aspects, keywords, sentiments and summary over time.

- **Better data collection and preprocessing**: The number of user reviews is really small compared to useless ones. As the number of reviews can get really big, it is advisable to gather only a subset of only the most important reviews based on certain criteria such as readability, length, etc. for further analysis. As we go through the extracted sentences, it is rare to find a badly structured sentence as we have carefully preprocessed them. But errors cannot be avoided completely. There are still cases where the extracted sentences are badly structured, too long, or meaningless. By preprocessing these sentences better, the results of other steps such as summary can be improved.

- **Deployment and scalability with big data technologies**: Deploying a complex machine learning system in production is no trivial task as the production environment differs a lot from the development environment. A lot of the proposed methods can be performed in parallel such as aspect extraction, sentiment analysis, text summarization, etc. The performance of the system can greatly be improved by implementing big data technologies for distributed computing.

- **Customized visualization**: Even though Power BI offers us a quick and easy way to build beautiful interactive dashboards, there are still lots of constraints with the integration of Python and other more complicated technologies in the program. That is why building the dashboard using a more robust visualization tool such as Ploly Dash, Streamlit, etc. based on this prototype built in Power BI can be a great next step in developing a complete and powerful machine learning system.
- **Human-in-the-loop approach**: Human-in-the-loop (HITL) machine learning combines human and machine intelligence to create a continuous circle in which the algorithm is trained, tested, and tuned. With every loop, the machine becomes smarter as well as more confident and accurate. As our system is entirely unsupervised, there are lots of things that could be improved in every component by leveraging labeled data and supervised methods. An example would be our sentiment analysis model. Currently, the unsupervised model classifies the sentiment of each sentence quite well, but there are still wrong sentences and the model cannot get better over time. An idea would be letting the user relabel falsely classified sentences interactively on the dashboard. A state-of-the-art supervised sentiment classification model can then be trained on this dataset labeled both by the unsupervised model and our users to deliver much higher accuracy and continuously improve over time. Or during text summarization, we can rate the quality of the extracted sentences and allowing the system to replace bad sentences with better ones.
- **Application in other domains**: Although the unsupervised methods developed in this study were applied only to video game reviews, they should be applicable to other domains where online reviews and opinionated text are prevalent such as product, app, restaurants, hotel reviews, tweets, etc. Further evaluation is needed to confirm this claim.

Bibliography

[1] Opinion summarization methods: Comparing and extending extractive and abstractive approaches. In: *Expert Systems with Applications* 78 (2017), S. 124–134

[2] Amplayo, Reinald K. ; Lapata, Mirella: *Informative and Controllable Opinion Summarization*. 2021

[3] Angelidis, Stefanos ; Lapata, Mirella: *Summarizing Opinions: Aspect Extraction Meets Sentiment Prediction and They Are Both Weakly Supervised*. 2018

[4] Bowman, Samuel R. ; Angeli, Gabor ; Potts, Christopher ; Manning, Christopher D.: *A large annotated corpus for learning natural language inference*. 2015

[5] Bražinskas, Arthur ; Lapata, Mirella ; Titov, Ivan: *Unsupervised Opinion Summarization as Copycat-Review Generation*. 2020

[6] Campello, Ricardo J G B. ; Moulavi, Davoud ; Sander, Joerg: Density-Based Clustering Based on Hierarchical Density Estimates. (2013), S. 160–172

[7] Chu, Eric ; Liu, Peter J.: *MeanSum: A Neural Model for Unsupervised Multi-document Abstractive Summarization*. 2019

[8] Clement, J: *Global video game market value from 2020 to 2025*. – Statista

[9] Clement, J: *Number of peak concurrent Steam users from January 2013 to May 2021*. – Statista

[10] Clement, J: *Number of games released on Steam worldwide from 2004 to 2021*. – Statista

[11] Clement, J: *Hierarchical Clustering / Dendrogram: Simple Definition, Examples*. – Statistics How To

[12] Devlin, Jacob ; Chang, Ming W. ; Lee, Kenton ; Toutanova, Kristina: BERT: Pre-training of deep bidirectional transformers for language understanding. In: *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference* 1 (2019), S. 4171–4186. ISBN 9781950737130

[13] Ester, Martin ; Kriegel, Hans-Peter ; Sander, Jörg ; Xu, Xiaowei: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, AAAI Press, 1996 (KDD'96), S. 226–231

[14] Explosion: *spaCy - Linguistic Features*

[15] Fan, Angela ; Grangier, David ; Auli, Michael: *Controllable Abstractive Summarization*. 2018

[16] Gambhir, Mahak ; Gupta, Vishal: Recent automatic text summarization techniques: a survey. (2017), S. 1–66

[17] Ganesan, Kavita ; Zhai, ChengXiang ; Han, Jiawei: Opinosis: A Graph-Based Approach to Abstractive Summarization of Highly Redundant Opinions. In: *Proceedings of the 23rd International Conference on Computational Linguistics*. USA : Association for Computational Linguistics, 2010 (COLING '10), S. 340–348

[18] Hutto, C. ; Gilbert, Eric: VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. In: *ICWSM*, 2014

[19] Kim, H ; Ganesan, K: Comprehensive review of opinion summarization. (2011)

[20] Lin, Dayi ; Bezemer, Cor P. ; Zou, Ying ; Hassan, Ahmed E.: An empirical study of game reviews on the Steam platform. (2019), S. 170–207. – ISBN 1066401896

[21] Liu, Bing ; Hu, Minqing ; Cheng, Junsheng: Opinion Observer: Analyzing and Comparing Opinions on the Web. In: *Proceedings of the 14th International Conference on World Wide Web*. New York, NY, USA : Association for Computing Machinery, 2005 (WWW '05), S. 342–351. – URL `https://doi.org/10.1145/1060745.1060797`. – ISBN 1595930469

[22] Liu, Yinhan ; Ott, Myle ; Goyal, Naman ; Du, Jingfei ; Joshi, Mandar ; Chen, Danqi ; Levy, Omer ; Lewis, Mike ; Zettlemoyer, Luke ; Stoyanov, Veselin: *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019

[23] Mikolov, Tomas ; Chen, Kai ; Corrado, Greg ; Dean, Jeffrey: *Efficient Estimation of Word Representations in Vector Space.* 2013

[24] Nallapati, Ramesh ; Zhai, Feifei ; Zhou, Bowen: *SummaRuNNer: A Recurrent Neural Network based Sequence Model for Extractive Summarization of Documents.* 2016

[25] Panagiotopoulos, George ; Giannakopoulos, George: A Study on Video Game Review Summarization. (2019), S. 36–43. ISBN 9789544520588

[26] Pennington, Jeffrey ; Socher, Richard ; Manning, Christopher: GloVe: Global Vectors for Word Representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* Doha, Qatar : Association for Computational Linguistics, Oktober 2014, S. 1532–1543. – URL https://aclanthology.org/D14-1162

[27] Pontiki, Maria ; Pavlopoulos, John: SemEval-2014 Task 4 : Aspect Based Sentiment Analysis. (2014)

[28] Reimers, Nils ; Gurevych, Iryna: Sentence-BERT: Sentence embeddings using siamese BERT-networks. In: *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference* (2020), S. 3982–3992. ISBN 9781950737901

[29] Ruder, Sebastian ; Peters, Matthew E. ; Swayamdipta, Swabha ; Wolf, Thomas: Transfer Learning in Natural Language Processing. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, 2019, S. 15–18

[30] SAS: *Data Visualization: What it is and why it matters*

[31] Suhara, Yoshihiko ; Wang, Xiaolan ; Angelidis, Stefanos ; Tan, Wang-Chiew: OpinionDigest: A Simple Framework for Opinion Summarization. In: *CoRR* abs/2005.01901 (2020). – URL https://arxiv.org/abs/2005.01901

[32] Tian, Yufei ; Yu, Jianfei ; Jiang, Jing: Aspect and Opinion Aware Abstractive Review Summarization with Reinforced Hard Typed Decoder. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (2019), Nov. – URL http://dx.doi.org/10.1145/3357384.3358142. ISBN 9781450369763

[33] Vajjala, S. ; Majumder, B. ; Gupta, A. ; Surana, H.: *Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems.* O'Reilly Media, 2020. – ISBN 9781492054054

[34] Wikipedia: *ROUGE (metric)*

[35] Williams, Adina ; Nangia, Nikita ; Bowman, Samuel R.: *A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference.* 2018

[36] Xu, Dongkuan ; Tian, Yingjie: A Comprehensive Survey of Clustering Algorithms. (2015)

[37] Yauris, Kevin ; Khodra, Masayu L.: Aspect-based summarization for game review using double propagation. In: *Proceedings - 2017 International Conference on Advanced Informatics: Concepts, Theory and Applications, ICAICTA 2017*, URL http://ieeexplore.ieee.org/document/8090997/, 2017, S. 1–6. – ISBN 9781538630013

[38] Zhao, Chao ; Chaturvedi, Snigdha: *Weakly-Supervised Opinion Summarization by Leveraging External Information.* 2019

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „— bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] — ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen.“

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: ____________________

Vorname: ____________________

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Entwicklung eines Machine-Learning-Systems zur aspektbasierten Sentiment-Analyse und Textzusammenfassung von Videospiel-Rezensionen auf Steam

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original