Applying Ranking Techniques for Product to Games

BY

EDWIN FONG CHIOK HOONG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

In partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Perak Campus)

JANUARY 2016

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: _____
_____
_____

Academic Session: _____

I _____

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

_____ _____

(Author's signature) (Supervisor's signature)

Address:

_____

_____ _____

_____ Supervisor's name

Date: _____ **Date**: _____

.

Applying Ranking Techniques for Product to Games

BY

EDWIN FONG CHIOK HOONG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

In partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Perak Campus)

JANUARY 2016

DECLARATION OF ORIGINALITY

I declare that this report entitled "**Applying Ranking Techniques for Product to Games**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____

Name : _____

Date : _____

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Dr. Ng Yen Kaow who has given me the opportunity to work in a natural language processing project. He also provided me a lot of guidance and feedback throughout the project. It is my first step to establish a career in this field. A million thanks to you.

I would like to thank my friends for supporting me and providing me ideas and feedback for this project. Finally, I also would like to say thanks to my parents for their continuous love, support and encouragement.

ABSTRACT

The video games industry has become a multi-billion-dollar global industry in recent years. There are plenty of game producers and publishers competing with each other to release their games to the video games market. Today, video games market has become more competitive than before. In order to survive in this huge competition, games companies must make correct decisions when producing games that able to gain profit. Recognizing trend is one of the ways that allows games company to do so by producing games that are able to attract as many video gamers as possible. There is an abundance of freely available game reviews and user discussions on the internet. From them, game producers can discover how various aspects of their games are being received. A new field of research called *aspect ranking* allows the automatic discovery of such knowledge from the reviews. The aim of this project is to adapt techniques in the field to produce a reviews analysis system that is aimed at helping game companies analyze large amount of online reviews in order to identify how they can improve their games.

The reviews analysis system includes five stages. The first stage extracts game reviews by crawling the relevant sites on the internet. The second stage extracts aspects from the text by discovering dependency relations between words in a sentence. Next, the system filters the candidate aspect list to clear some false aspects. Aspect ranking is the third stage of this system. In this stage, HITS algorithm is used to rate each aspect in the candidate aspect list by their importance. Using these scores, the system can rate the overall performance of a game by aggregating their highly rated aspects, weighted by the importance of the respective aspects. The overall performances of games computed by the system compare favorably with game rankings in online websites; this lends credibility to the system's reliability. Encouraged by this success, the system further employs the aspects list in various ways to facilitate the analysis of a game's performance with respect to other games. It is hoped that game producers will find these features useful in understanding the strengths and weaknesses in their games. Last but not least, these analysis methods are made available to the user through a web interface, which also provides visualization of the results.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Motivation and Problem Statement

The video games industry is very large. Millions of video games have been developed and released in the market. Each new game has to compete with an enormous number of games that are already available in the game market. Due to the vigorous competition, game publishers typically carry out reasonable analysis before they release their game to the market. One of the possible analyses is to find out the current trend in successful games. To accomplish this, the use of aspect discovery or aspect ranking could be very useful to analyze the secret formula behind successful games.

By looking through the latest trend of video games, firms can easily target current gamer by releasing a game that is attractive to current society. Besides developing games, firms can also formulate their advertising and marketing strategy to sell their games to a broader audience. For example, they can focus their advertisement by highlighting on a specific aspect that are attractive from the audience such as stunning graphics, interesting storyline or others.

As mentioned, the game industry is highly competitive. To ensure the success of a game, game producers typically put in substantial amount of effort in analyzing the game market before developing or releasing a game.

To understand the situation of the current game market, game producers need to identify the current trend in the gaming market, and analyze the behavior and interest of current gamer. However, it is difficult to analyze gamer behavior. As an example, user reviews on games can be used to analyze and understand their behavior but it takes a lot of time if the game producer read each of the reviews one after another. There could be millions of reviews on the game review website, forums and game selling website. Therefore, for the purpose of summarizing of the reviews, a game aspect ranking system can be introduced to automatically identify and rank each game aspect. By using the result from the system, game producer could make important decision to increase their sales.

1.2 Background Information

The video gaming industry is a multi-billion-dollar global industry – its market permeates the society, being played by people of all ages, on devices ranging from all types of consoles, PCs, to mobile devices. In 2013, video game companies sold 160 million units of video games, collecting about USD 21.53 billion of revenue overall. Gartner estimated that worldwide video game sales will reach USD 111.1 billion by 2015. In addition, a study done by Economists Incorporated (2014) shows that the interactive entertainment software industry achieved a growth at an annual rate of approximately 10 percent in the United States. At the same period, the overall US economy only grew by 2.4 percent.

In the last decade, the game industry has changed a lot in terms of both the gaming platforms as well as the target audience. Gamers are not the same anymore and their behavior is different compared to the past. People now increasingly play a more active role in the development of the games, instead of just consuming them. This is due to the advancement in computer and communication technology. Different aspects of gaming have been introduced such as *modding*, streaming, cosplay and competitive gaming. Among them, live streaming of video games is now the latest trend of the gaming world. Millions of people stream their game online to share their gaming experiences and tips. Due to this reason, live streaming has become a serious business in gaming market. As an example, one of the most popular live streaming service provider, Twitch, claimed to have more than 2 million concurrent viewers at the same time and had the revenue of more than USD 1 billion in 2014 according to a study by SuperData Research. Recently, popular video content provider Youtube saw the opportunity in video game live streaming and decided to enter the gaming market by introducing Youtube Gaming. As a result, the gaming market now provides much broader and diverse opportunities and challenges than ever before.

Besides that, mobile games have now become one of the most important gaming platform for gamer and publisher. More than one third of monthly spending among digital video games is mobile game. However, only a few game publishers managed to claim any significant shares of the revenue; their games occupy the top positions in Apple's App Store's chart. These publishers are believed to have identified the interests of their audience, allowing them to release games that able to attract a large number of gamers. Example of games that manage to attract a large number of player are Flappy

Bird, Dots, Cut The Rope, Angry Bird, 2048, Candy Crush and many others. From the example, all of them share a same characteristic, which is “simple to play”. The simplicity is the component that attract gamer to download and play the game. Therefore, understanding the secret behind the key market trends and drivers of change is very important in a market where game companies have access to ranking information. For example, this firms should find out what drives the gamer to spend money on gaming and what are the opportunities are there in the market.

Due to the reason above, the ability to discover the important aspects to create successful product has become more and more important for firms. They aim to analyze the secret components and patterns of successful games before working on creating a new game. This type of aspect discovery, or aspect ranking, is very important for game producers. However, it is very difficult to find a ranking system based on aspect ranking.

On the other hand, there is an abundance of online data for the evaluation of products. The rapid growth of e-commerce have resulted in many online shopping websites, many of which allow customers to rate, or even write reviews on their products. Marketing researchers have been able to mine this online data, using sophisticated methods to transform them into insights for firms to better develop and market their products.

There are likewise, an abundance of online user reviews on video games, and it is interesting to see how well current techniques apply to these data. Currently, we know of no existing research which specifically mines such data. There are some online websites that rank games; however, the methods which they apply are unsophisticated. For example, one of the most popular games review aggregator website Metacritic will only go through selected publication sites instead of using user reviews to rank their games. According to their website, the staffs manually go through each site and assign score for them for ranking. The application of more sophisticated product ranking techniques to games may improve the depth of such websites.

1.3 Project Scope

This project will create and develop a game aspect analysis system to discover aspect, rank, and analyze the aspects of a given game from its reviews. First of all, the game aspect analysis system will extract consumer reviews from several websites that

contain games reviews. Next, the system is to automatically identify aspects from the reviews that have been extracted. After this, the system will perform aspect ranking. That is, the importance of each aspect will be inferred, and a score assigned to the according to its inferred importance. The credibility of the inferred ranking is tested through incorporating them in a system which rank games, and then by comparing the resultant ranking to well-established ones. The aspect list is then used for various analytics, to allow the evaluation of a game in comparison to other games.

1.4 Project Objectives

The proposed system is based on the discovery and ranking of aspects in game reviews. There are 3 main uses for the aspect ranking.

1. To find out trends in successful games

A lot of people wonder is there a key behind games or any pattern that make game more addictive than another. (Edoardo, 2014). For example, an interesting trend has emerged that has made some games successful. This trend is based on a simple principle which is to build a game without an end; people will keep playing just to improve their score and to beat friends' scores. From this example, simplicity and infinity are the patterns that we always found in mobile games. Plenty of popular games on the market fit this observation such as Flappy bird, 2048, Candy Crush, Temple Run, Fruit Ninja and many others. Since all of them share a similar formula, they enjoy similar success in terms of revenue. Therefore, firms should analyze the component and pattern of successful games before they develop their own game.

2. To find the interest of target audience

For over a decade, video games have evolved into a multi-billion-dollar global industry. Each video game required millions of dollars to develop and market. To make their investment worth it, they have to gain as much profit as possible. Therefore, they have to develop a game that can attract the most number of audiences. Games typically appeal only to a segment of the population, that is, a target audience. Firms want to target a larger audience rather than a small specific group to increase their game sales. In order to accomplish this, firms can collect information to get to know more about

their audience. It is important to identify and their gaming behavior to aid in developing the game

3. To develop marketing and advertising strategy

There are hundreds of thousand video games published over the years. In order for a game to be successful, it has to compete with all of the games available in the market. Therefore, to attract users to play their game instead of others, marketing and advertising strategy also play an important role. By understanding the behavior and the interest of the gamer, firms can determine which specific aspect or benefit of their game should be highlighted to impress the gamer.

1.5 Impact, Significance and Contribution

As mentioned, video game producers and publishers must find a way to compete with other video game company. The project will produce a series of analysis which allows producers to decide which type of game to invest in before the video games are produced or released to the market. One of these is the study of current trends in game preference. These trends from current gamers or players are obtained from games reviews on the internet. The project also allows game producers to evaluate how the games they have published are being received, by performing similar analytics on games review. This will help them identify the aspects which their games are weak or strong in. Similar analytics can also help producers better understand specific aspects by referring them to the games which are strong in specific aspects. This knowledge will allow game companies to make better business decision.

1.6 Report Organization

This report consisted of 6 chapter below: s

Chapter 1: Introduction

Chapter 2: Literature Review

Chapter 3: System Design

Chapter 4: Methodology and Tools

Chapter 5: Implementation and Testing

Chapter 6: Conclusion

CHAPTER 2: LITERATURE REVIEW

The growth of internet and online shopping has caused many users to use the internet to express and share their opinions about a particular product. Valuable information on product receptions, user habits and preferences, market trends, etc. can be mined from these online sources. In particular, companies are interested the positions of their products among users, as well as how they may improve products. This has intensified researches in two areas of data discovery from internet sources:

1. Product ranking

   Product ranking determine which product is the best and most popular among consumer.

2. Product aspect ranking

   Product aspect ranking determine which aspect of the product are more important among consumer.

2.1 Product ranking

Product ranking ranks products based on the opinion of the consumer. The better the product is, the higher its rank. It can be used to help consumer in making purchasing decision by providing them the most popular product voted by others. Consumer opinion can be extracted through comment and star rating. Other properties such as helpfulness vote and review posting date can also be used to apply to the ranking.

2.1.1 Star rating

Star rating is a way to get consumer opinion of a product. This rating is indicated by number of stars; the highest ranked product is the one with the most stars. Consumer usually gives a rating to a product after they bought it based on the quality of the product. However, there are issues in collecting and aggregating reviews from different sources:

1. Different sources have different rating scales (Example are 1-5 stars, 0-10 stars and etc.)
2. Bias rating

3. Fake rating and reviews

The most popular aggregate review score is the average rating per item. However, according to a study (Hu, Pavlou, and Zhang (2006, pp. 324-330), this is not always the best way to measure the true quality of a product. One study (McGlohon, Glanceand Reiter, 2010) listed that some issues in ranking product including compare reviews from different sources, filtering out unreliable reviews and finding true quality of product. In order to solve the problem, two main categories of models which are statistical and reweighting are analyzed. Statistical models include average rating, median rating, lower bound on normal confidence interval, lower bound on binomial confidence interval and average percentile of order statistic. Re-weighting model include filter out anonymous reviews, filter out non-prolific author and weighting authors according to maximum likelihood. Results show that average rating had the best accuracy with the rest are not far behind except median rating which had only about 48%. Possible future improvements include leveraging additional features of reviews, cleaning dataset for plagiarism or spam and leveraging other data sources.

2.1.2 Comments

Besides star rating, product reviews usually consist of comments from consumer. There are a lot of works in retrieving opinion from the comment from consumer in product reviews. Consumer normally commented on the quality of product in sentences. Each sentence may contain negative and positive opinion from the consumer. Huang and Lin (2013, pp.184-190) proposed a product ranking system using opinion mining techniques to help users find the favorable product. This product ranking system use product reviews and product popularity to calculate product scores. The system framework consisted of 7 components. As a reference, the detail of system framework is explained below.

1. First, it downloads the product HTML pages from the selected websites.
2. Next, it extracts the review and information of the product including how many people think the review is helpful, product rating, review title, release date and review text.

3. The system splits the text of the product review into sentences and produces POS tags such as noun, verb, adverb, adjective and etc.

4. The system then will start to determine the polarity of opinion words in the sentence to identify sentence polarity. In order to extract opinion words a dictionary based approach using Orientation Prediction (Hu, M., and Liu, B., 2004) is used. 15 common adjectives are then defined for both positive and negative sets in the seed list. The system will find the synonyms and antonyms of words in the seed list and iterates this step until no new synonyms or antonyms are being added into the seed list. Next, the system will perform opinion strength calculation, inverse document frequency which reduce effect of common adjectives and enhance important adjectives, and determine the degree of adverbs to weaken or enhance the adjective strength. Sentence polarity is then calculated using opinion strength, IDF and degree of adverbs.

5. The system will build the product XML files.

6. The system will rank the products using the average polarity of reviews, popularity weight and weight of product release month.

7. The final part of the system will be user interface.

2.1.3 Helpful votes and posting date

In order to obtain more accurate result, weight can be applied to the product review factors for product ranking score calculation. Examples of information that can be used are helpful vote, review posting date and any other relevant properties. Helpfulness vote are cast by posterior customers and it determine the importance and helpfulness of a review. From the total number of helpful votes and number of total votes, the helpfulness of review can be obtained. Furthermore, age of review is also another thing to consider. Newer reviews are usually written by consumers that have read some of the previous review. The newer reviews should receive more weight as they are more influential to potential customer. Zhang, Cheng, Liao and Choudhary (2011) build a filtering mechanism to preprocess review using the Support Vector

Machines to improve product ranking result. The proposed system consisting of three stages:

1. Filter out sentences that is unrelated to product quality
2. Derives weights for a review based on its helpfulness votes and review age
3. Calculate the product's overall ranking score.

2.2 Product Aspect Ranking

Product aspect ranking is to help consumers and firms to identify important product aspects by ranking the aspects of each product. Product aspect can be described as the product specification attributes or any component for a product. Therefore, consumer can make wise decision on purchasing by paying more attentions on the important aspects while firms can put more efforts in improving quality of important aspects to enhance product reputation. In order to gain the consumer opinions, comments in review are used to rank the product aspects.

To identify important product aspects from online consumer reviews, Yu, Zha, Tang, Wang and Chua (2011) suggested that important aspects can be determined according to two observations.

1. Important aspects of a product are usually present review comments by large number of users
2. Consumer opinions on the certain important aspects will greatly influence their overall opinions on the product.

According to Zha, Yu, Tang, Wang and Chua (2014), they proposed a product aspect ranking framework which is able to automatically identify the important aspects of products from reviews of users. This product aspect ranking framework consists of 3 main steps:

1. Product aspect identification

   Consumer review consists of pros and cons review, free text review or both.

For pros and cons review, the system extracts the frequent noun terms as study shows aspects are usually noun phrases (Liu 2005). For free text review, frequent noun terms are extracted but leverage from pros and cons is used to eliminate noises if available.

2. Sentiment classification on product aspects

   A sentiment classifier which can be SVM, Naïve Bayes or Maximum Entropy Model is trained to determine consumer opinions. This approach is a supervised learning approach while other way is to use lexicon-based approach which is easier to implement but relies heavily on the quality of the sentiment lexicon. A study from Wu, Xu and Li (2011) also showed that supervised approach offered comparable performance using Support Vector Machine.

3. Probabilistic aspect ranking algorithm.

   Overall rating in each of the review is obtained based on the weighted sum of the opinion of specific aspects.

On the other hand, there is also another method using double propagation algorithm. According to study of Li, Guan, Tang and Chen (2012), they proposed a method using double propagation to extract product features and double propagation based linear regression for ranking product features. Double propagation (Qiu, Liu, Bu and Chen, 2009) exploits the relations between sentiment words and product features that the sentiment words modify to extract product features. Zhang, Liu, Lim and O'Brien-Strain (2010, pp. 1462-1470) suggest that double propagation are mainly used to extract medium-size corpora and perform very well. However, this method introduced a great deal of noise and has low precision for large size corpora while intend to miss important product features for small size corpora. Double propagation assumes that features are nouns/noun phrases and opinion words are adjectives. In large corpora, this method may extract a lot of unrelated nouns/noun phrases which are not features and therefore low precision is obtained. The other problem that introduced through

double propagation is that for certain domains, some important features lack of opinion words that modify them. To deal with all of the problems, the authors proposed a novel method to mine features. The features extraction part still adopts the double propagation method to populate feature candidates whereas the product ranking consists of two improvement based on part-whole relation patterns and a “no” pattern to find features that double propagation unable to find.

CHAPTER 3: SYSTEM DESIGN

There are five main steps in this system as shown in figure 3.0 which is consumer review extraction, aspect extraction, aspect ranking, game score prediction and data visualization.

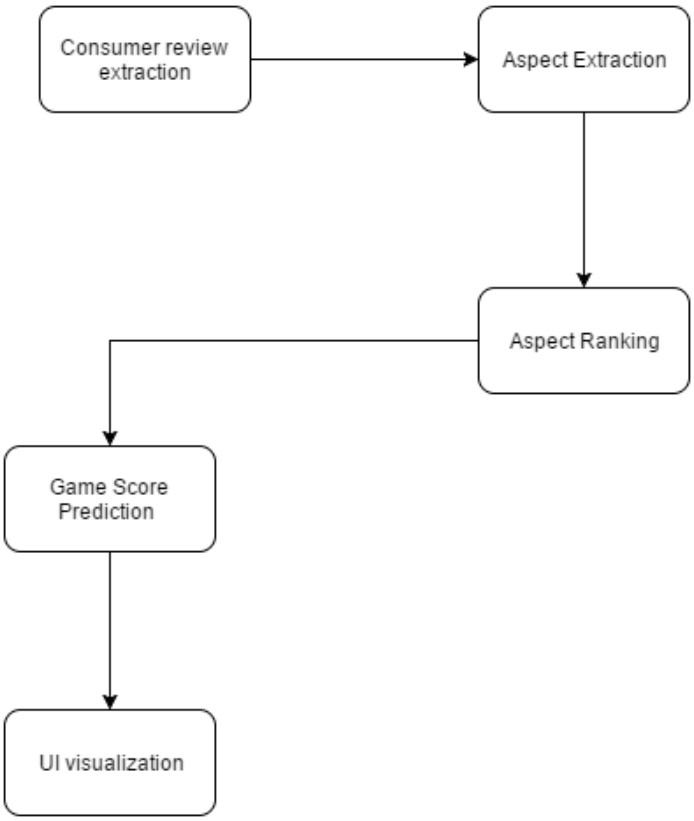


Figure 3.0 Flow of the whole system.

3.1 Consumer review extraction

Collecting consumer reviews from review websites is the first step of this system. Examples of the websites include metacritic.com, gamestop.com, ign.com, amazon.com and other video game review websites.

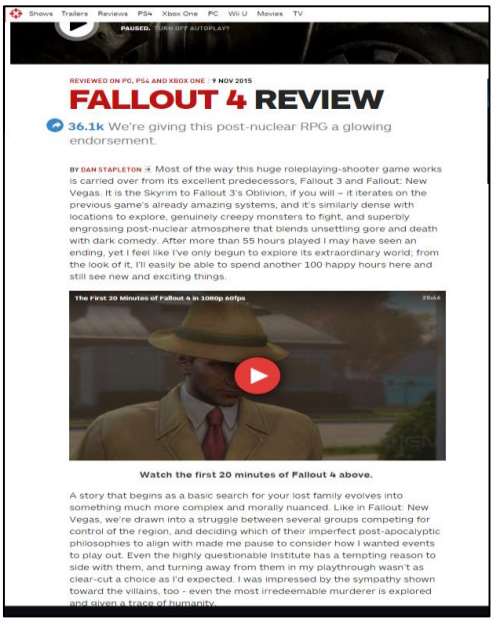


Figure 3.1: Example of review website

In order to obtain a large amount of reviews from the internet, a tool called Scrapy is used in this system. Scrapy is a web crawling framework that allows data extraction from review websites in this system. Since it is written in Python, this part of the system will be written and developed in the Python programming language. Scraping reviews from the internet may take a very long time and large bandwidth. To overcome this issue, a Linux cloud server is deployed in order to scrape data from review websites.

Game reviews are spread out all over the internet. This increases the difficulties to obtain all the sources for those game reviews. Fortunately, Metacritic.com had already aggregate games review from most of the game review websites. Therefore, links to those sites can be found on the Metacritic.com website. The way to implement Scrapy will be explained next.

First, Scrapy can be installed directly using Python pip. Then, a new Scrapy project is created by using a Linux command “scrapy startproject crawlReview”. After that, a new directory call “crawlReview” consisting some starter code is automatically

created after the previous command is issued. In order to define how it will scrape a certain site, spiders will be created inside the "spiders" folder. Spiders are Scrapy python classes that define how to perform the crawl and how to extract data from the page itself. For the purpose of this system, there are a total of 3 spiders created. The first spider crawls and extracts the games list and their URL in Metacritic.com. By using each game URL in Metacritic.com extracted earlier, the second spider will crawl each URL and extract the URL of each game review that is contained in the website. The last spider is then responsible for crawling each game review sites and extract game review as well as other relevant information. After done crawling, everything is saved into a database for storing.

3.2 Aspect Extraction

To perform aspect extraction from the extracted review, there are two steps to extract explicit aspects:

1. Exploiting the dependency relations
2. Filtering the candidate aspect list

3.2.1 Exploiting the dependency relations

Between opinion words and their opinion targets, there are many syntactic relations that linked them together. Furthermore, it is observed that opinion words often modify or describe on their opinion targets (Hu and Liu, 2004). An example of such relation is "The software is amazing." where we can clearly identify "software" is the aspect in the sentence and "amazing" is the opinion word that modifies the aspect. Therefore, dependency relations between words in a sentence can help us to find aspect in game reviews. Furthermore, dependency is a one-to-one connection for words in a sentence and word are connected through the knowledge of grammar.

To exploit this kind of relation, they can be identified by the help of a dependency parser. Dependency parser is capable of linking words in a sentence by dependency relations. This method not only extracts aspect but it also simultaneously performs the extraction of both aspect and the sentiment words of the aspect. Therefore, a dependency parser from Stanford parser is used in this project to identify the dependency relations of opinion and opinion targets to extract possible list of aspect-sentiment pair.

The latest version of Stanford Parser required at least Java version 8 or above and a high amount of memory to run. Stanford Parser is currently included together with other natural language analysis tools in the Stanford CoreNLP package which can be downloaded from the official website of Stanford CoreNLP (http://stanfordnlp.github.io/CoreNLP/#download). The total file size is about 536MB and included POS tagger and sentiment analysis tools that will be used in the later stage of the system.

After successfully downloaded, the zip file is unzipped to extract the file. The zip file included some demo java file, jar file and bash shell scripts to run in Linux command line. Before writing any program, the dependency parser is first tested to make sure the correct outcome is produced before being the development. For testing purposes, the prewritten bash script file named “lexparser.sh” is used on a newly set up Linux machine in VirtualBox. To run it, the command used is “./lexparser.sh file.txt” where file name is the only argument to pass in through command line.

During the first implementation of the following steps, an error had occurred from Java regarding the out of memory exception. This problem occurred because there is not enough memory for the parser to run. There are two solutions to solve this problem. In the first solution, we can increase the heap size of Java virtual machine by passing an argument to indicate how much memory should be allocated to it. In this system, the heap size for the program is increase to 2048MB by modifying the original lexparser.sh and add the argument “-mx2048m” when calling java to run as shown in figure 3.2.1.1.

```
edwin@ubuntu:~/stanford-parser-full-2015-04-20$ cat lexparser.sh
#!/usr/bin/env bash
#
# Runs the English PCFG parser on one or more files, printing trees only

if [ ! $# -ge 1 ]; then
  echo Usage: `basename $0` 'file(s)'
  echo
  exit
fi

scriptdir=`dirname $0`

java -mx2048m -cp "$scriptdir/*:" edu.stanford.nlp.parser.lexparser.LexicalizedParser \
 -outputFormat "penn,typedDependencies" edu/stanford/nlp/models/lexparser/englishPCFG.ser.gz $*
edwin@ubuntu:~/stanford-parser-full-2015-04-20$
```

Figure 3.2.1.1: Increase the heap size of JVM.

The second solution is to increase the maximum length of sentence with the argument "-maxLength". This solution restricts the maximum length of a long sentence and the long sentence will be cut into multiple sentences if it is longer than the maximum length allowed. By default, Stanford Parser differentiates each sentence by period mark if nothing is specified. This solution may not be ideal as words in a sentence will not be linked together if they belong to a different sentence. However, it is still very useful when there is not enough memory available in the system to allocate anymore. In my test, the first solution is chosen as there is enough memory to allocate for it to run without errors.

To run the whole system easily, a program is written in Java to include Stanford Parser. In this system, it is decided to implement this on Eclipse since it supported Java. The same technique is also used to increase the max heap size of the system to prevent the system to fail as mentioned previously. The use of the Stanford Parser as an API in Java is quite straightforward. First, the package "edu.stanford.nlp.parser. nndep.DependencyParser" is imported to the program, then a "DependencyParser" object is created which contains the function to predict the dependency between word in the sentence. This package includes the Neural Network Dependency Parser which is a super-fast transition-based parser. During the development, an outdated dependency parser called Lexicalized Parser is used as shown in figure 3.2.1.1. However, the performance of this parser is very slow especially when there are a large number of sentences to be parsed. After some research, Neural Network Dependency Parser which is newly added to the Stanford Parser is decided to replace the previous parser due to its high speed and increased accuracy according to Chen and Manning (2014).

Next, a simple description of relations between words in a sentence is outputted after being parsed by the Stanford Parser as shown in figure 3.2.2.2 below.

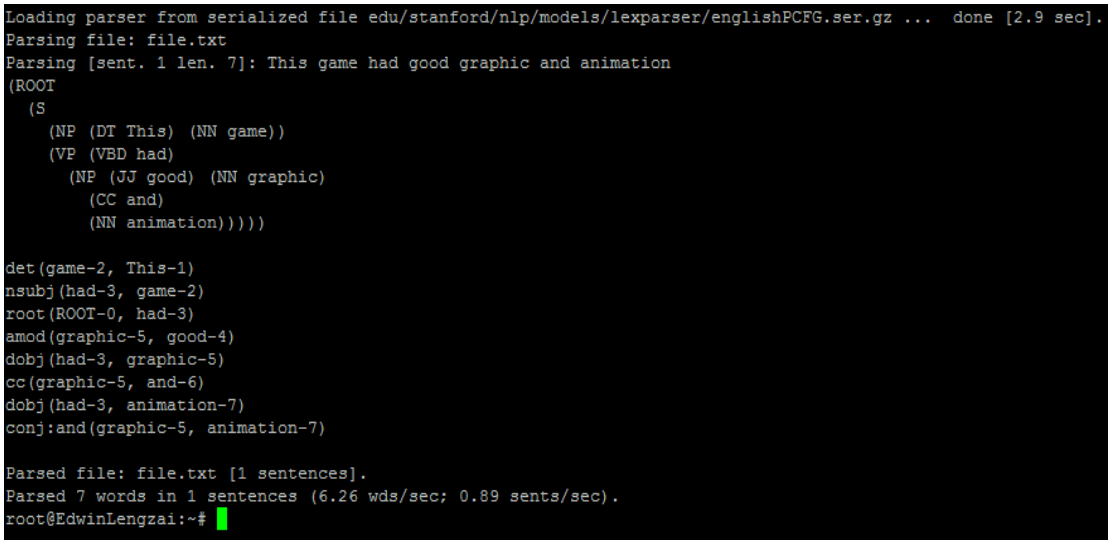


Figure 3.2.1.2: Example output of Stanford Parser.

In figure 3.2.1.2, the output of the Stanford Parser shows tree representation of dependencies that each word in a sentence will have a certain relation with one of another. By default, Stanford Parser outputs the universal Stanford dependencies that these dependencies map straightforwardly onto a representation of directed graph. Universal Stanford Dependencies (De Marneffe, M. C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., & Manning, C. D., 2014) is a new project for developing cross-linguistically consistent treebank annotation for many languages. There is a total of 40 universal relations in the latest version of Universal Stanford Dependencies. Universal Stanford Dependencies provide a universal inventory of guidelines and categories to help facilitate consistent annotation of similar constructions across many different languages. It provides a representation of grammatical relations between words in a sentence to the user that is designed to be easily understood. There are always three triplets in the Universal Stanford Dependencies, that is, the name of the relation and the two words in sentence. For example, “amod(society, modern)” where “amod” is the name of the relation and the words inside the bracket are currently connected based on this relation which is “society” and “modern”.

In Figure 3.2.1.2, “amod” is denoted as adjectival modifier which is any adjective phrase that is capable of modifying any noun or noun phrases. In this figure, the words that are in the “amod” relation are “good” and “graphic” where “good” is an

adjective and “graphic” is a noun. In addition, the word “graphic” is likely to be an aspect and “good” is a sentiment term that describe the “graphic”. Due to this characteristic, it could help to discover sentiment-aspect pair in the games review. Therefore, this relation is included in the system.

Besides “amod” relation, there are also other relations which are modifiers such as “neg”, “advmod”, “det” and etc. Like “amod”, the “det” relation is one of the relations that usually appeared in our text. This is the relation between a determiner and the head of a noun phrase. For example, “Which book do you prefer” where “book” is the object and “which” is the determiner. As a result, this relation is not suitable for our search of aspect and sentiment as it does not contain any sentiment in this relation and it is unlikely to be an important aspect. Next, “advmod” relation was known as adverbial modifier of a word which is usually a word that adverb served to modify the meaning of a word. For example, “Genetically modified food” where the word “Genetically” is an adverb and modify the word “modified” in the sentence. Due to this characteristic, it is also possible to be used to discover aspect in game review as well.

On the other hand, “conj” describes a relation between two words connecting with each other by a coordinating conjunction. An example of such coordinating conjunction are “and” and “or”. This is also a very important clue for extracting aspect from the review text. For example, if “graphic” is likely to be an aspect, then “animation” is also equally likely to be an aspect as well. This method is not only used to find aspect but also can be used to find sentiments that are connected with a known sentiment by a coordinating conjunction. For example, “Fast and simple gameplay” where “Fast” and “simple” are both adjective and opinion words. Therefore, if we know that “simple” is an opinion word, we can also determine “simple” as an opinion words. Besides “conj()”, there is also another relation that work by making use of a coordinating conjunction which “cc()” relation. It is a relation of the first conjunct and coordinating conjunction delimiting another conjunct. Example of such relation is “The game had good graphic and animation” and the word “graphic is in “cc()” relation with the word “and”.

Other than that, there are a group of relations which consists of “nsubj”, “nsubjpass”, “csubj” and “csubjpass” which belong to a more general relation “subj”, or also known as subject. “Nsubj” is a nominal subject which is a nominal phrase that is the syntactic subject of a clause. Example of such relation is “The graphic is good”

where the word “graphic” is in a “nsubj” relation with the word “good”. Another example would be “Ronny defeated Kenny” where the word “Ronny” is in a “nsubj” relation with the word “defeated”. “Nsubjpass” is similar to the former but it is a phrase which is the syntactic subject of passive clause. On the other hand, “csubj” relation is a clausal syntactic subject of a clause.

In figure 3.2.1.2, there is a relation called “dobj” which is defined as direct object relation. It normally describes the object that the verb phrase points to. For example, “They win a lottery” where the relation is dobj(gave, rise). This relation is also possible for aspect discovery as there is always a noun phrase. In addition, a “root” relation points to the root of the sentence where the root is the word that are not a dependent and it is a governor.

How well are the performances of these relations in extracting aspects in games? A series of test is carried out to test the outcome of each relation. To perform this test, about 20+ reviews of a particular game is downloaded and saved into text files. After this, all those reviews were parsed by the Stanford Parser using the step stated above. Then, results from all sentence of each review were all saved together to a temporary text file. After this is done, I tried to extract lines of the results that contain a particular relation using the Linux command grep. For example, “grep amod review.txt” is typed in command line to extract all lines that have “amod” relation. Furthermore, this step is repeated for all relation that is possible to discover aspect as stated as above. At the end, it is discovered that “amod” and “conj” relation able to provide a list of sentiment-aspect pair while the rest does not yield a good result. For this reason, only “amod” and “conj” relations are selected.

In the previous step, opinion words are involved in order to find the aspect when finding sentiment-aspect pair. Besides opinion words, there is also another clue that can be used to find aspects. A relation proposed by Zhang and Liu (2010) is used in this system. This relation is known as a part-whole relation. According to the paper, it is a useful indicator for finding aspect if the class concept word is known. In the case for this project, we can assume the word “game” and the name of the game as the class concept. An example of this relation is “graphic of the game” where we can identify “graphic” as part of the “game”. In order to recognize this relation, there are several patterns of part-whole relation that exist in phrase as shown below:

1. NP + Prep + CP
2. CP + with + NP
3. NP CP or CP NP
4. CP + verb + NP

where NP is noun/noun phrases, CP is class concept phrase and prep is preposition.

In the 1st pattern above, noun phrases is the “part word” and class concept phrase contains the “whole word”. They are also connected by a preposition such as “of”, “in” and “on”. An example of this pattern is “animation of the game”. Next, in the second pattern, CP and NP are connected by a “with”. For example, “game with sandbox” where “game” is the class concept and “sandbox” is likely to be an aspect. The third pattern involves the noun phrase and class concept phrase to form a compound phrase. An example of this pattern is “open-world game” where “open-world” is an aspect and “game” is the class concept. In the fourth pattern, the verb is between the class concept phrase and noun phrase. The verb that can be included here is “has”, “have”, “include”. “contain”, “consist”, and “comprise”. An example of this pattern is “The game contains excellent storyline” where we know that “game” is the class concept here and “Excellent storyline” is the aspect that we are looking for. This part-whole relation concept should be added into the system as it is well fit to the criteria of game reviews.

On the other hand, adjective phrases or opinion words usually indicate aspect such as “good graphic”, “nice animation” and so on. Besides this, there is also another way to indicate aspect such as “no” pattern. For example, consider “no multiplayer” and “no setting”, where the word “no” is followed by a noun, and is likely to be an aspect.

3.2.2 Filtering the candidate aspect list

The first step in filtering candidate aspects is to remove non-noun aspects. Due to the fact that an aspect is always a noun and those words that are non-noun are not likely to be an aspect, so this will be a clue for us to find aspect. Due to this reason, we need to remove any non-noun words from the candidate aspect list. However, before we can remove those non-noun words, we need to identify which word is noun and which word is not. Therefore, the knowledge of part of speech is needed to solve this problem. In the interest of categorizing each word into their particular part of speech, Stanford Part of Speech (POS) Tagger is used for this purpose. Since Stanford POS tagger is already included in the CoreNLP package, it is not necessary to download the separate package again.

```
import edu.stanford.nlp.tagger.maxent.MaxentTagger;

public class Temp {
    private static MaxentTagger tagger = new MaxentTagger(
            "postagger3/models/english-left3words-distsim.tagger");
    public static void main(String[] args){
        tagger.tagString("This game had good graphic and animation");
    }
}
```

Figure 3.2.2.1: Example code to implement Stanford POS Tagger in Java

To implement Stanford POS tagger in Java, three steps is necessary as shown in figure 3.2.2.1:

1. Import MaxentTagger
2. Declare a MaxentTagger
3. Use tagString method from MaxentTagger to process the string to perform POS tagging.

```
This_DT game_NN had_VBD good_JJ graphic_JJ and_CC animation_NN
```

Figure 3.2.2.2: Output of Stanford POS Tagger

After the code is run, the output is as shown in figure 3.2.2.2. Each word is marked with the part of speech at the end of each word. For the noun, words will be marked by “NN”, “NNS”, “NNP” and “NNPS” (Comp.leeds.ac.uk, 2015). NN is denoted as singular common noun such as “thermostat”, “investment”. NNS is denoted as singular proper noun where proper noun is the name used for place, people and thing such as “Liverpool”, “Malaysia”. An NNP is a plural common noun and an NNPS is a plural proper noun where both are the same except the former is singular and the latter is plural. For the java program to know which word is a noun, it can be easily done by reading the end of each word with word that ends with those four marks as noun. Next, all non-noun aspect will be eliminated and the remaining noun aspects will be included in a new candidate aspect list.

Furthermore, words that have less than 3 alphabets are eliminated from the candidate aspect list because they are unlikely to be an aspect. Although words that are unlikely to be an aspect or an unimportant aspect will be ranked very low in the later stage, filtering the aspect in this stage can ensure the overall aspect list later to be a lot cleaner and easier to read.

Next, the system will also remove pronouns from the candidate list. Pronouns are words that are substitutes for noun/noun phrases. Examples of pronouns are “he”, “her”, “his”, “others”, “something” and etc. As stated in section 3.2.2.1, aspects that are not noun are already filtered out but pronouns in this case are not removed. Due to the fact that most part of speech tagger does not consider pronoun as a single class itself and consider them as a noun. Hence, a separate list of pronoun is compiled and used to get rid of pronouns from being considered as noun in the list. To compile this list, list of pronouns is retrieved from a website (Esldesk.com, 2015) and it contains a total of 74 pronouns in the whole list.

In addition, a list of stop words is also added to the list together with the pronoun. Removing stop words is a common practice in natural language processing. Examples of stop words may be common, short function words such as “the”, “is”, “at”, “which,” “on” and so on. A universal list that contains all the stop words does not exist. Therefore, a list of 661 words of stop words is added together with the 74 pronouns with the goal to filter as much unwanted words as possible before the candidate aspect list is processed and ranked in the further stage. In Java program, the removal is quite

straightforward where any aspect exists in the list is remove from the array of aspect together with its sentiment that describe it.

After removing all the unwanted aspect, a problem still existed in the list which may affect the entire result in the later stage. This problem is the existence of singular form and plural form of a single individual aspect in the list. For example, the words “graphics” and “graphic” is the same similar word and the only different are one of them is a singular and one is plural. Without any modification and processing, both of the words will be recognized as 2 different words by the program later. This will cause a very serious problem because it affects the ranking list by allocating different score to each word where both of them supposed to be counted as a whole. In order to solve this problem, we can turn all plural words in the list to the original singular form. However, there is a major challenge to accomplish this because there are 3 types of plural words which is regular plurals, near-regular words, irregular words. For regular forms, the plural is suffixed to the end of most nouns. For example, a noun with sibilant sound will end with “es” in plural form such as “kisses”, a noun end with “o” will end with “es” in plural form and the rest of the word will end with the most common “s”. This form of plural is easy to handle, however, near regular and irregular form is a lot harder to turn into singular such as “children” and “child”. To solve this problem, part of the java code from Github repository for singularization is used. The author had already defined all the possible rules to perform singularization. After singularization is performed, the list now only contains singular noun aspect.

3.3 Aspect Ranking

After getting the necessary aspects and information, the system will rank the aspects based on their relevance and importance. Therefore, an assumption is made that aspects that are important and relevant will rank high in the list whereas unimportant and irrelevant aspects will be ranked low in the list. In this system, HITS algorithm is applied to for this purpose.

The HITS algorithm (Kleinberg, 1999), abbreviated from Hypertext-Induced Topic Search, was originally designed for search engines to rank webpage. It works in a way that gets the most useful, relevant and helpful results of a particular query from the user. Since most useful webpages may not have the keyword for the query and pages are not descriptive enough, so HITS algorithm is used to solve this problem. It is a link analysis algorithm that finds authoritative results.

The HITS algorithm assigns every page an authority score and hub score, where the authority score estimates the value of the content of the page and the hub score estimates the value of its links to other pages. According to Kleinberg (1999), hubs and authorities exhibit a mutually reinforcing relationship. It is stated that a good hub is a page that links to many useful authorities while a good authority will be referenced by many good and useful hubs.

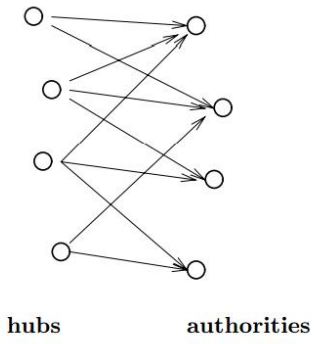


Figure 3.3.1: Relations between hubs and authorities

In figure 3.2.3a, it is shown that hubs may link to multiple authorities and authorities may be referenced by multiple hubs. In addition, an iterative algorithm is required to make the idea above happen. It works by iteratively or repeatedly maintains and updates the authorities and hub weight for each page. Weights for each page should be non-negative as well. The equation to calculate authorities and hub are:

$A(i) = \sum_{(i,j)\in E} H(j)$ (Equation 1)

$H(i) = \sum_{(i,j)\in E} A(j)$ (Equation 2)

Where A = authority score, H = Hub score, E = set of directed edges,

According to the idea of this iterative algorithm, it first starts with each node having a hub score and authority score of 1. Next, the authority score of all nodes will be calculated based on Equation 1 shown above. In Equation 1, the hub score of each node that have a link to it are summed up and become the authority score of that particular node. Then, it will now update all the hubs score based on Equation 2. In Equation 2, the authority score of each node that is referenced by it are sum up and become the hub score of that particular node. Since the authority score is calculated first, therefore the hub score will be calculated based on the new authority score. Furthermore, normalization will also be applied so that the iterative process will converge. The normalization is done by dividing each hub score by the square root of the sum of the squares of all hub scores as well as dividing authority score by the square root of the sum of the squares of all authority score. If no normalization is applied to the value of both score, they will become larger with each iteration and become an infinite iterative process.

Iterate(G,k)
  G: a collection of n linked pages
  k: a natural number
  Let z denote the vector $(1, 1, 1, \ldots, 1) \in \mathbf{R}^n$.
  Set $x_0 := z$.
  Set $y_0 := z$.
  For $i = 1, 2, \ldots, k$
    Apply the $\mathcal{I}$ operation to (x_{i-1}, y_{i-1}), obtaining new x-weights x'_i.
    Apply the $\mathcal{O}$ operation to (x'_i, y_{i-1}), obtaining new y-weights y'_i.
    Normalize x'_i, obtaining x_i.
    Normalize y'_i, obtaining y_i.
  End
  Return (x_k, y_k).

Figure 3.3.2: Pseudocode of HITS algorithm

From the formulas and pseudocode, it is observed that authority score estimates the importance of the content of a webpage and hub score estimates the worth or value of the page pointed to the other page. Due to this relationship, it is possible to be applied to this project scenario because we already gained three clues or pattern which is the target of opinion words is likely to be an aspect, part-whole relation and “no” relation. This is say so because an aspect is important if it is modified by many features, exist in part-whole patterns and “no” patterns. Therefore, the concept fit the HITS algorithm perfectly as it also consists of a mutual enforcement relation that can be easily included into the HITS algorithm framework.

According to Zhang, Liu, Lim and O'Brien-Strain (2010), HITS algorithm can be applied to obtain feature relevance ranking. Therefore, in this project, aspects will act as authorities and aspect indicators will act as hubs to model the HITS algorithm framework. Based on the idea of HITS algorithm, an aspect that is modified by many opinion words or in part-whole relation and “no” relation, is more likely to be an important feature whereas aspect indicator such as opinion words that modify a lot of aspects is likely to be a good aspect indicator.

In short, this system will run HITS algorithm with the initial hub scores and authority scores set to 1. Then, the power iteration is used to update both scores until they converge. Finally, the authority scores of all the aspects are retrieved for the next process.

In HITS algorithm, each aspect can only appear in 1 authority node. Due to this reason, the frequency of the aspect is not taken into account yet. Frequency is also a

very important criterion for an important aspect. Therefore, a new score will be calculated based on the frequency of the aspect together with the authority score that have been retrieved from previous HITS algorithm process. The formula for the new score is calculated as below:

$$S = A(f) \log freq(f)$$

where S = final score of aspect f,

$A(f)$ = final authority score of aspect,

freq(f) = frequency of aspect.

Even though the new scores have been calculated, the system still has much remaining work to do. This is because some high frequency general nouns are usually highly ranked in the candidate aspect list such as “thing” and “people”. This is inevitable because the system includes the frequency of each aspect that makes it easy for common words like these to be ranked in a high position. Besides that, the HITS algorithm also favors these kinds of general nouns since it promotes aspects that are modified by many opinion words, and high general nouns are very frequently described by an opinion word such as “good thing”, “excellent thing”, “nice people”, “wonderful time” that often appear in game reviews. Since the game review contains too many of opinion expressions that contain non-target terms, a solution must be applied to prevent the general nouns from polluting the aspect ranking list.

To solve this problem, a list of common words sorted with how frequent it is in common texts can be compiled. To compile this list, I downloaded a list from a website that contains the ranking of top 1500 most frequent nouns (Talkenglish.com, 2015). As shown in figure 3.2.3c, the 1500 frequent nouns come together with the frequency of it. By making use of this frequency, the list is sorted based on their frequency. Therefore, the ranking of 1500 frequent nouns is produced and ready to be used.

On the other hand, another way of generating this kind of corpus is proposed by Xu, L., Liu, K., Lai, S., Chen, Y., & Zhao, J. (2013). According to their method, they generated a small domain-independent general noun corpus from some large web corpora to cover as much frequently used general nouns as possible. First, 1000 most frequent nouns in Google-1-gram were selected as general noun candidates. Furthermore, all nouns in the top three levels of hyponyms in four WordNet synsets

which are “object”, “group”, “person” and “measure” into the general noun corpus. The following steps are taken due to the idea that a term is more general if it is located in a higher level in the WordNet hierarchy. WordNet is a massive lexical database of English where nouns, verbs, adverbs and adjective are grouped into sets of cognitive synonyms. At the time the paper is published, they were able to create a 3071-word English GN corpus.

The idea here is, a common word that is highly ranked in the common words list compiled will reduce the final score of the aspect in the aspect list significantly, while low ranked common words should not affect the final score as much as the high ranked one. Therefore, I propose the following formula for this purpose:

$$FinalScore = \frac{Score}{N/R}$$

where N = number of common words in the common word list,

R = ranking of common words in the common word list.

Finally, every aspect in the new list now has a final score. The list will be sorted again based on their final score and the ranking for the aspects is produced. Outcome of the result is shown in table 3.3.1 below. Compared to the preliminary result previously, the new result is obtained by processing more than 10000 reviews and the most important aspect is calculated. All sentiment score and aspect score will be saved in a text file for later use as well.

3.4 Game Score Prediction

This stage of the system will predict the score of each game using the ranked list of aspect obtained from the earlier stage. Each game obtained from the game review will be scored separately. To score each game, sentiment and aspect for each game is extracted from their respective game reviews using the similar technique mentioned in section 3.2. However, the aspect ranking will not be performed here anymore.

Next, sentiment analysis will be performed on each sentiment-aspect pair extracted. Sentiment analysis is performed by using Stanford Sentiment Analysis tools that is already included CoreNLP package used in the previous steps. This additional tool is one of the latest tools implemented by the Stanford NLP Group. The underlying technology of the following is based on a new form of recursive neural network which builds on top of the grammatical structures. According to Socher, R. *et al*., this model is expected to outperform all previous models and was able to classify a single sentence for positive and negative up to 85.4%. Furthermore, this model is also the only model that is able to recognize the effects of contrastive conjunctions and negations. Thus, it is used to predict the sentiment of each game reviews.

```
if (text != null && text.length() > 0) {
    Annotation annotation = pipeline.process(text);
    for (CoreMap sentence : annotation
            .get(CoreAnnotations.SentencesAnnotation.class)) {
        Tree tree = sentence
                .get(SentimentCoreAnnotations.SentimentAnnotatedTree.class);
        mainSentiment = RNNCoreAnnotations.getPredictedClass(tree);
    }
}
return mainSentiment;
```

Figure 3.4: Example code of performing sentiment analysis

As shown in figure 3.4, each sentence will be processed separately and form a tree structure representing the grammatical relationship between words in the sentence as mentioned in section 3.2.1. This is needed because the sentiment model from Stanford NLP used these relationships as one of the criteria to predict the sentiment such as the conjunction of multiple sentiments. In the case of this system, the sentence passed in to the function is the aspect-sentiment pairs extracted from reviews. At the end of the code shown in figure 3.4, the result of the sentiment returned will be either 1, 2 or 3 where 1 is negative, 2 is neutral and 3 is positive.

If the sentiment of the aspect is negative, the total score is subtracted by the product of the sentiment score and the aspect score. If the sentiment of the aspect is positive, the total score is added by the product of sentiment score and the aspect score. Otherwise, the total score is added with only the aspect score itself.

Earlier, the sentiment score and aspect score is calculated using the HITS algorithm. Therefore, this sentiment score and aspect score will be used to determine how much aspect and sentiment to be added into the current total score. The program will then iterate through all the sentiment-aspect pairs obtained from all the reviews of the particular game and repeat the process of adding or subtracting the score using the technique mentioned above.

To evaluate the quality of the ranking obtained, I compared it to the ranking of games in Metacritic.com. Unfortunately, the rankings of a few games turned out to be in discrepancy with their rankings accorded by Metacritic. After further tests and examinations, it was found that these discrepancies correspond very well to the cases where the games have too few reviews. The artifacts are hence, very likely due to the lack of statistical significance. For this reason, a decision is made where only games with at least 10 reviews will be used in this comparison.

After the games with few reviews are removed, the ranks obtained using my method compared very favorably with the ranking given in Metacritic.com. More precisely, pairwise game rankings according to the system is almost identical to that according to Metacritic.com. When tested with 30 2015 games, the top 50% of the games ranked by the system achieved scores between 80 to 96 in Metacritic.com score, whereas the lower 50% of the games ranked by the system scored lower than 80 in Metacritic.com. However, some exceptions or outliers remained.

There are several possible factors that may have contributed to these discrepancies. The first possibility is that the games released in different year may also have been reviewed under different criteria. For example, a game in year 2004 such as “Half Life 2” was among the best games in that year, with state-of-the-art graphics, amazing gameplay, and groundbreaking concept. Due to this reason, the game is given a very high score by Metacritic.com and other game reviewers. However, if a similar game is released in 2016, it would most likely not be received with similar reviews, even though it contains all the elements and aspects of “Half Life 2”. That is, a game

which looked good in 2004 may not look as good in 2016 anymore. Therefore, a game should only be compared or ranked among games in that particular time. Furthermore, games with different genres also should be compared separately because each genre caters to a different audience and different point of view. When comparison is restricted to only games of the same genre and of the same year, I observed no discrepancy between the system's ranking and that of Metacritic.com.

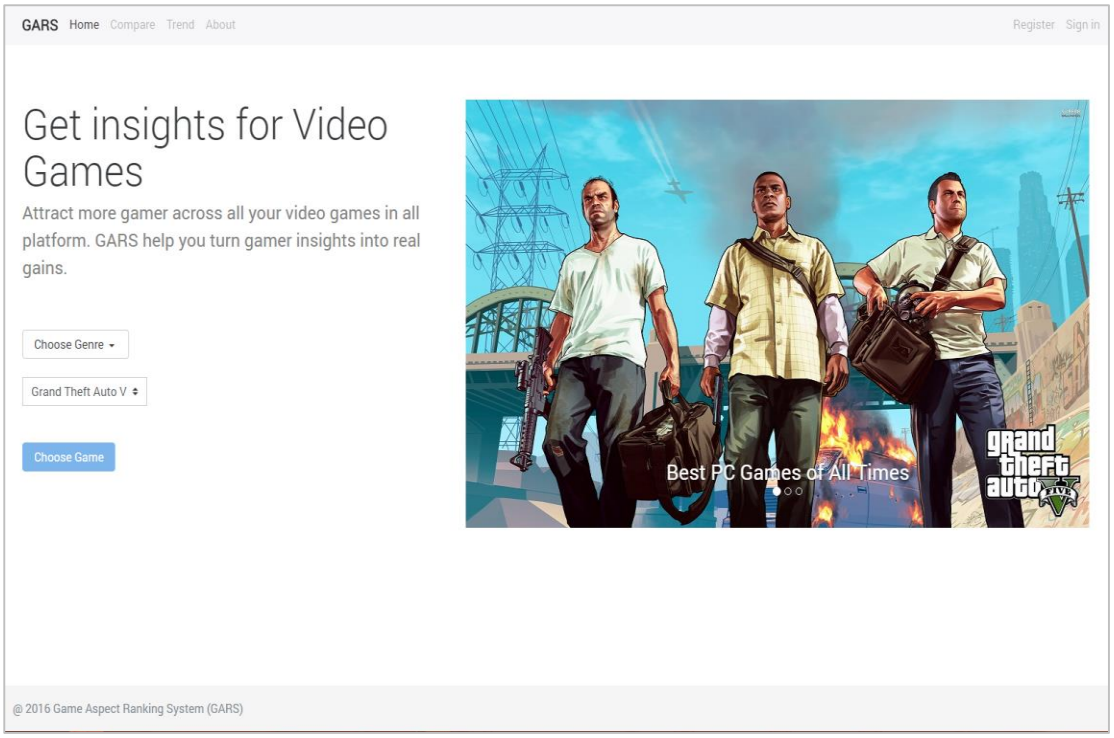
Lastly, there is a technical issue which is fairly obscure: as a sales strategy, some companies frequently update their games with new expansion or DLC (Downloadable Content) where new content is added to the original game such as additional storyline besides original storyline or new character or skills. In some case, these DLC or expansions will be reviewed as well. These game variants skewer the results since their reviews will typically be focused solely on the aspects that have been updated.

With all these factors considered, the ranking of the system achieved very good consistency with that from Metacritic.com. This implies that the aspect ranking system is correct to some extent.

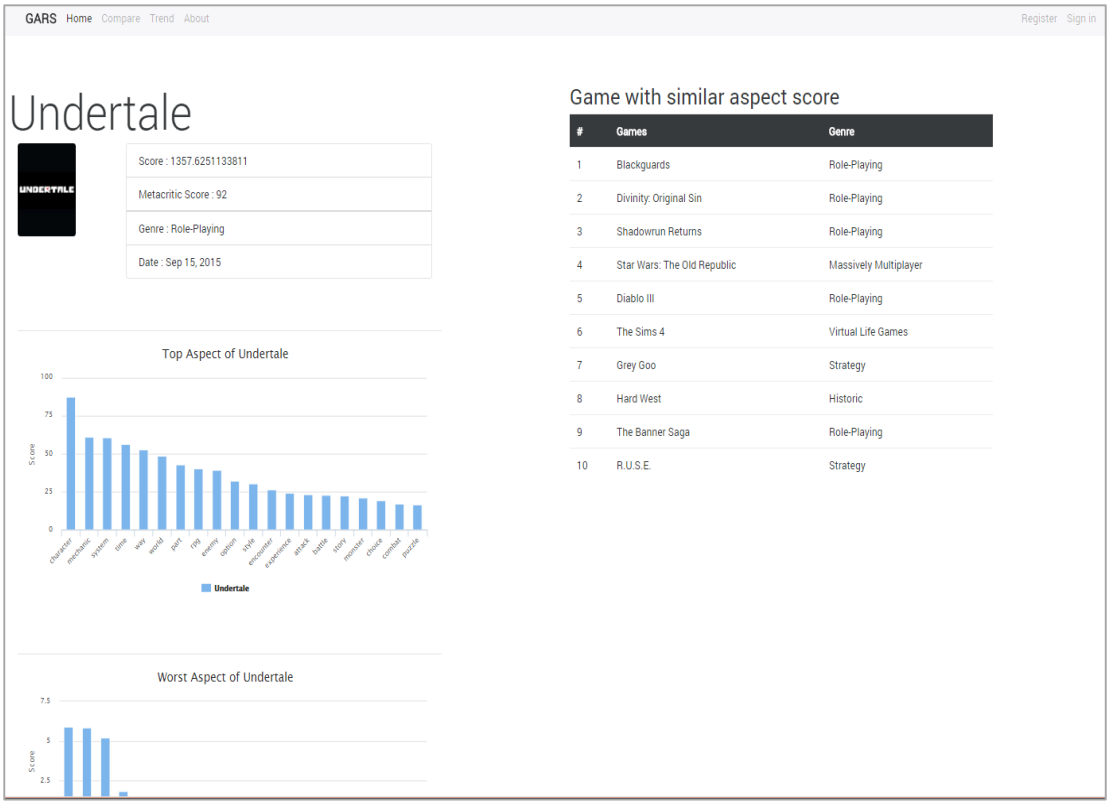
3.5 UI visualization

Finally, I consider the use of the discovered aspects and their ranking for analytics purpose. For example, the aspects can be presented in a visual manner for user examination. Or, a chart can be plotted to show how game aspects have changed through the time.

Several of these uses are provided through a web UI, created using PHP. The backend is linked to the data and results obtained from the Java program earlier. The first page of the website is the homepage as in figure 3.5.1. It allows user to choose the genre they want and filter the game list according to the genre selected. After user selected the games, the current games will be analyzed, and the user is directed to the result page.



Example 3.5.1: Screenshot of the homepage.



Example 3.5.2: Screenshot of result page (Part 1)

Next, the result page contains the analysis of the game including the game top aspect and worst aspect. Both the top aspect and worst aspect is presented using a

column chart so that user can analyze what is the strength of their game and their weakness. Furthermore, some details of the game are also displayed such as game cover, score, Metacritic.com score, game and date of release.

Furthermore, the website will display the top 10 games that are similar to the game being analyzed. To accomplish this, the distance between every game in the database with the game being analyzed is calculated. Any method to measure distance such as Euclidean Distance should work in finding the similarity between 2 games. In this system, a pre-written class for PHP is used to calculate the distance where each aspect and their aspect score is used to calculate the similarity between 2 games.

Lastly, a web chart is presented to show the allocation of aspect in a particular game to analyze which aspect contribute a lot of credit to the game as shown in figure 3.5.3.

Example 3.5.3: Screenshot of result page (Part 2)

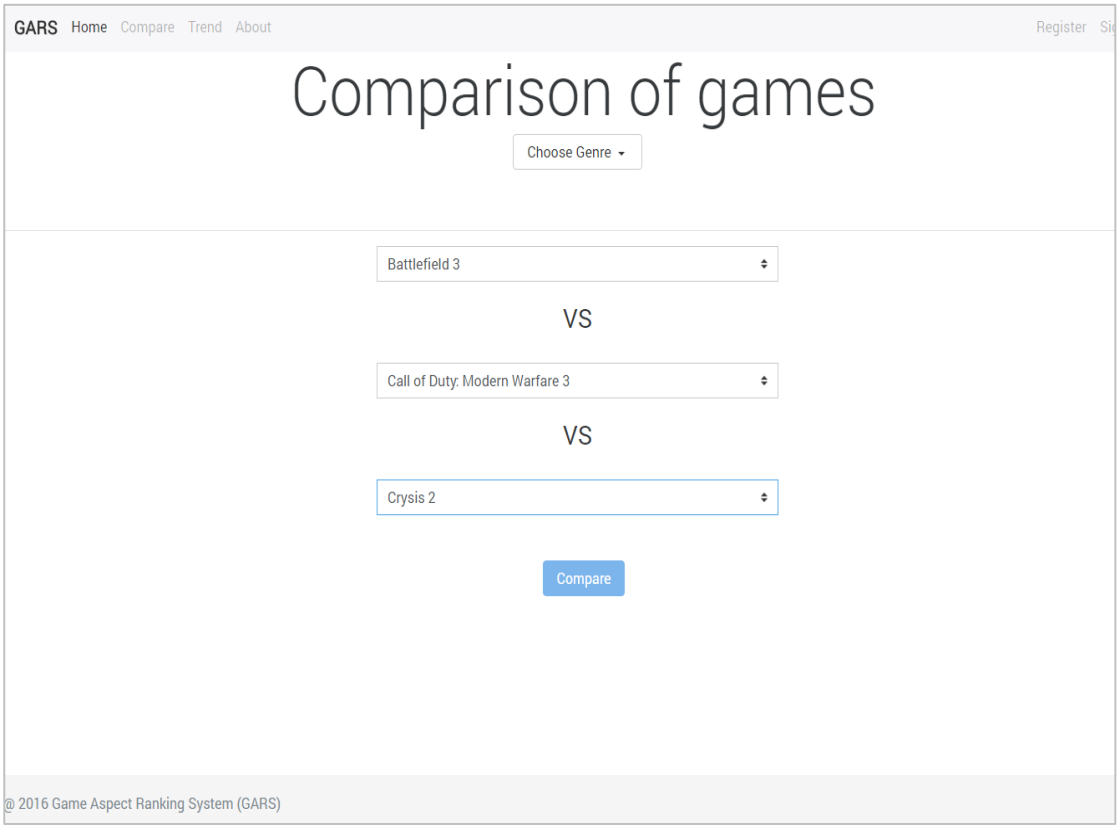


Figure 3.5.4: Screenshot of compare page

A separate page allows user to compare games of same genre. The user first chooses a genre to filter the game list. After user click compare, the result will be show in next page.

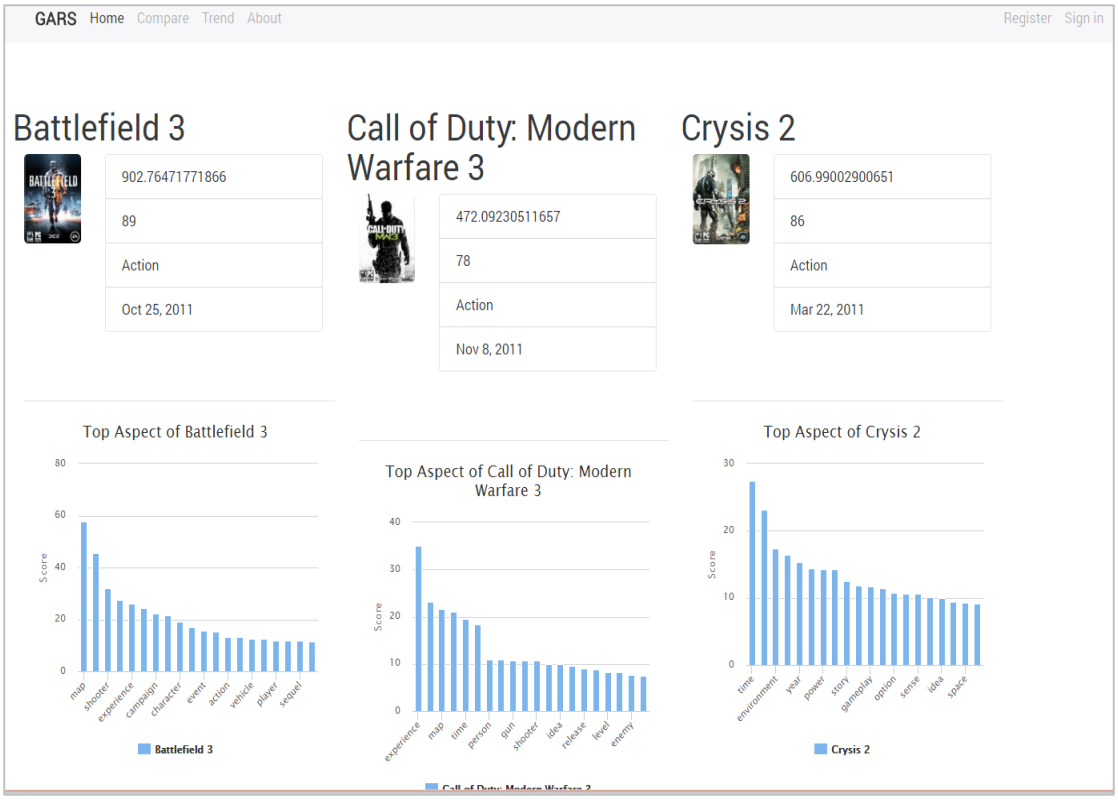


Figure 3.5.5: Screenshot of comparison result page.

In the comparison result page, 3 games are shown side by side allowing user to compare the strength and weakness of each game. According to the current screenshot, it is shown that the score provided by the system fits the score Metacritic.com perfectly. After some testing, in most of the time, the result of the system is the same with Metacritic.com as long as the genre and year of release of the game is the same.

Figure 3.5.6: Screenshot of trend page (Part 1)

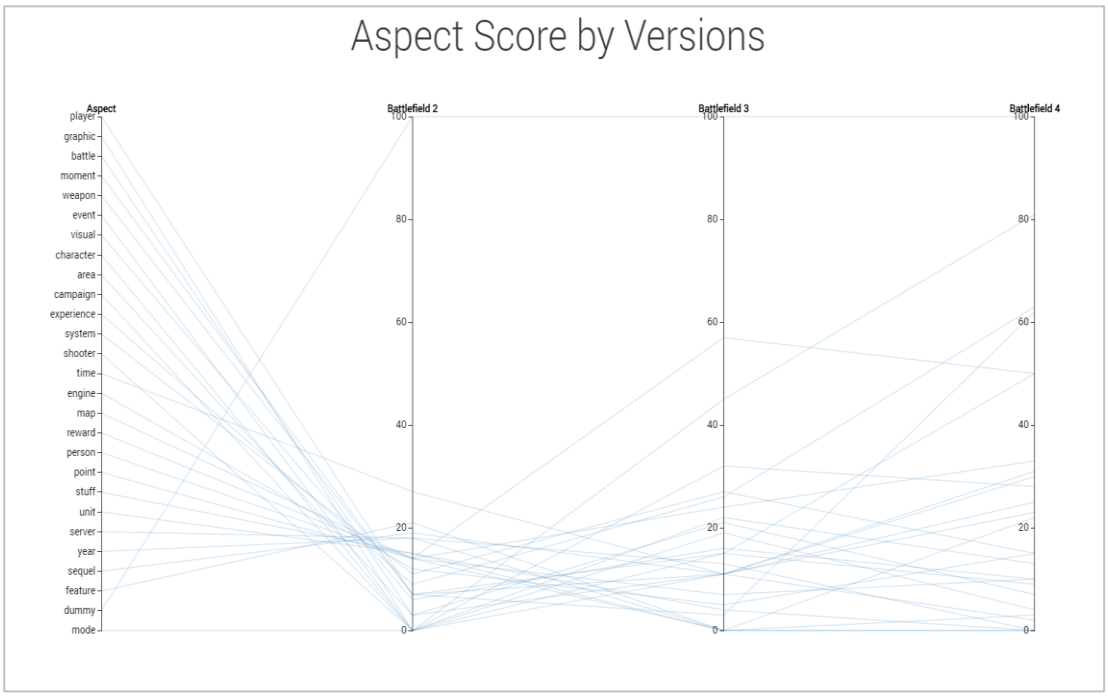


Figure 3.5.7: Screenshot of trend page (Part 2)

In the trend page of the website, users are given a visualization of the trend of the game, across different years or across different versions of the same game.

There are two graphs in the page. The first graph shows how the game performs in recent versions so that user can check whether the game series is improving or getting worse by observing the score of the games in different versions. The second graph allows user to perform more detailed analysis by looking at how each individual aspect performed in each version. For example, if the user implemented or improved a particular aspect from the old version, he/she will be able to check whether the aspect they had worked on has been positively received by the reviewers.

3.6 Implementation Issues and Challenges

There is currently one difficulty in getting good games reviews. Some reviews sites allow anyone to post their opinion on it. While these sites usually have massive amount of discussions, it is difficult to obtain aspects from these reviews, discussions and comments. One example of this type of website is steamcommunity.com. Therefore, the games reviews to be extracted for this system will try to avoid these websites as much as possible.

3.7 Timeline

A Gantt chart is produced in order to estimate time required for each task and make sure the project can complete on time.

Task Name	Duration	Start Date	End Date	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
1.1 Requirement gathering	3 days	26/10/2015	28/10/2015							
1.2 Study project background	4 days	29/11/2015	1/11/2015							
1.3 Determine problem statement	2 days	2/11/2015	3/11/2015							
1.3 Detemine project motivation	2 days	4/11/2015	5/11/2015							
1.4 Determine project scope and objective	2 days	6/11/2015	7/11/2015							
2.1 Literature review	4 days	8/11/2015	11/11/2015							
2.2 Review existing system	4 days	12/11/2015	15/11/2015							
3.1 Determine Methodology	1 day	16/11/2015	16/11/2015							
3.2 System Design	6 days	17/11/2015	23/11/2015							
FYP 1 Submission	NA	23/11/2015	23/11/2015							
Presentation and Poster Submission	NA	9/12/2015	9/12/2015							

Figure 3.7.1: Timeline of Final Year Project 1

Task Name	Duration	Start Date	End Date	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
1.1 Requirement gathering	6	18/1/2016	23/1/2016							
1.2 Study project background	8	24/1/2016	31/1/2016							
1.3 Determine problem statement	4	1/2/2016	4/2/2016							
1.4 Determine project motivation	4	5/2/2016	8/2/2016							
1.5 Determine project scope and objective	4	9/2/2016	12/2/2016							
2.1 Literature review	10	13/2/2016	22/2/2016							
2.2 Review existing system	6	23/2/2016	28/2/2016							
3.1 System design	28	29/2/2016	27/3/2016							
4.1 Determine methodology	2	28/3/2016	29/3/2016							
4.2 Identify tools	2	30/3/2016	31/4/2016							
5.1 Implementation and Testing	10	1/4/2016	10/4/2016							
FYP 2 Submission	1	11/4/2016	11/4/2016							
Presentation and Poster Submission	1	20/4/2016	20/4/2016							

Figure 3.7.2: Timeline of Final Year Project 2 (Part 1)

Task Name	Duration	Start Date	End Date	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 1
1.1 Requirement gathering	6	18/1/2016	23/1/2016							
1.2 Study project background	8	24/1/2016	31/1/2016							
1.3 Determine problem statement	4	1/2/2016	4/2/2016							
1.4 Determine project motivation	4	5/2/2016	8/2/2016							
1.5 Determine project scope and objective	4	9/2/2016	12/2/2016							
2.1 Literature review	10	13/2/2016	22/2/2016							
2.2 Review existing system	6	23/2/2016	28/2/2016							
3.1 System design	28	29/2/2016	27/3/2016							
4.1 Determine methodology	2	28/3/2016	29/3/2016							
4.2 Identify tools	2	30/3/2016	31/4/2016							
5.1 Implementation and Testing	10	1/4/2016	10/4/2016							
FYP 2 Submission	1	11/4/2016	11/4/2016							
Presentation and Poster Submission	1	20/4/2016	20/4/2016							

Figure 3.7.3: Timeline of Final Year Project 2 (Part 2)

CHAPTER 4: METHODOLOGY AND TOOLS

4.1 Methodology

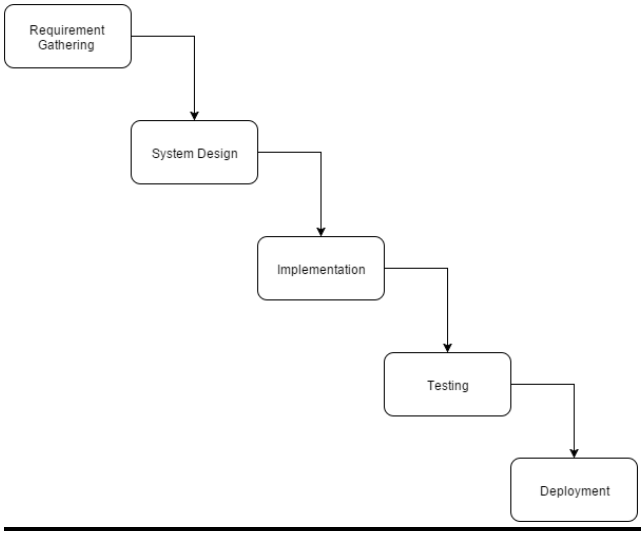


Figure 4.1: Waterfall model of this project

The system development methodology used in this project is the waterfall model. The reason that this development methodology is used is because it is simple to use and the system development process of this project also fits the flow of the waterfall model since our project does not have any phases overlap.

Stage 1: Requirement Gathering

In this phase, all possible requirement for this project is captured and documented. The existing system, method and technique are also studied and analyzed to help determine the requirement of this project.

Stage 2: Design

The requirement from previous phase is then studied to prepare a system design. Next, hardware and software tools are specified and the overall system architecture is designed based on the existing methods that are similar to this project.

Stage 3: Implementation

In this stage, the system will be developed based on the system design.

Stage 4: Testing

After developed the system, the different part will put together for testing.

Stage 5: Deployment

Once the testing is completed, the final result of the system is recorded.

4.2 Tools

To develop this system, different tools and programs are used. Below are the tools and programs used in this system:

4.2.1 Java

The Java programming language is selected to develop this system due to two main reasons. First, Java is an object-oriented computer programming language that allows application developers to run the compiled Java code on any platform that support Java without the need of recompiling. Second, several natural language processing tools that are used in this project are built and run using Java.

4.2.2 Eclipse

Eclipse is selected as an integrated development environment for Java programming. Eclipse is currently the most popular Java IDE among developers. Besides Java, other programming languages are supported through the use of plugins.

4.2.3 Stanford Parser

Stanford Parser (Chen, D., and Manning, C. D., 2014) is a natural language parser that works on the grammatical structure of a sentence. It will provide a simple description of grammatical relationships between words or phrases in a sentence. It is easy to use and to understand even for people without linguistic expertise when extracting textual relations. For example, it will determine which words are subject or object of a particular verb.

4.2.4 VirtualBox

VirtualBox is a powerful virtualization tool for enterprise and home use. VirtualBox runs on Windows, Linux, Macintosh and also supports many other operating systems. It helps to run virtual machine on top of a host operating system. VirtualBox is required in this project as a virtual machine is needed to be setup. Some resources and tools used in this project such as Stanford Parser needs to run on Linux system. My current system is running Windows operating system so VirtualBox is used to run a Linux system.

4.2.5 Stanford POS tagger

Same as Stanford Parser, Stanford POS tagger is a standalone package from Stanford natural language processing group. It is designed to assign part of speech to each word such as noun, verb, adverb, adjective and etc.

4.2.6 Scrapy

Scrapy is a web crawling framework that enables fast and simple data extraction from website. It is free and open-source. The data extracted is useful in variety of application including data mining, information processing as well as historical archival. It is written in Python and can be installed using the Python pip.

4.2.7 Microsoft Visual Studio 2015 Community Edition

Microsoft Visual Studio 2015 Community Edition is an IDE developed by Microsoft to develop computer programs, websites, web server and others. It is a free full functionality IDE. It supports a variety language such as PHP, Python, C++ and web technologies like HTML, CSS, Javascript. Due to this reason, it is used to code Python programs and PHP for web visualization purpose.

CHAPTER 5: IMPLEMENTATION AND TESTING

In this chapter, the system as mentioned in section 3 which describe the overall system design will be evaluated. There are 2 parts of the system where result is needed to be evaluated which is aspect ranking and game score prediction.

5.1 Result of Aspect Ranking

In order to observe the result, the program is executed where game reviews are used as the input. Before executing the ranking of aspect, aspect extraction is performed as stated in section 3.2. After aspect ranking is performed, a list of 8828 aspects is produced. A part of the result is shown in the table below.

Table 5.1.1: Result of Top 32 aspect obtained

1	Character	9	Mission	17	Battle	25	Area
2	Mode	10	Player	18	Adventure	26	Map
3	Puzzle	11	Time	19	Feature	27	Style
4	Experience	12	System	20	Point	28	Effect
5	Story	13	Unit	21	Version	29	Option
6	Level	14	Weapon	22	Action	30	Gameplay
7	Title	15	Element	23	Part	31	Mechanic
8	World	16	Enemy	24	Person	32	Shooter

No reference game aspect list can be found on the internet for comparison. Hence, it is hard to make comparison with any existing list. When look into the current list in table 5.1.1, common aspects are ranked very high in the list. Although these aspects are very common, if a game lacks these common aspects, the game is supposed to be less successful. Therefore, it can be deduced as common aspects are very important aspect. For this reason, it can be used to prove that the current list is accurate since important aspects are ranked high in the list. Next, the bottom of the list will be examined in table below.

Table 5.1.2: Result of Bottom 32 aspect obtained

1	sourcestone	9	defiance	17	valet	25	plu
2	spencer	10	reviewtopic	18	chateau	26	ban
3	yunkai	11	balentien	19	geese	27	discourse
4	holstein	12	nougat	20	witnessat	28	tobacco
5	upcomi	13	boner	21	oldlost	29	alcohol
6	ghale	14	timeshift	22	carball	30	milk
7	mating	15	wonk	23	writ	31	crazy
8	veer	16	jehovah	24	capitalism	32	conduct

In table 5.1.2, 32 aspects in the bottom of the list are shown. It is observed that all of them are misspelt words and random words that are not important. This again shows that the list is capable of separating good and bad aspect where good aspect are ranked high in the list and vice versa.

5.2 Result of Game Score Prediction

On the other hand, the result of aspect ranking can be used to perform another task which is game score prediction. Eventually, the game scores can be used to rank games as well. It is mentioned in section 3.2 where only games with the same year and genre can be used to reliably compare their scores. Therefore, the scores of several games generated by the system are compared with the Metacritic.com score.

Table 5.2.1: Result of game score prediction

Game Genre	Year Released	Game 1	Score given by Metacritic.com	Score given by system
Action (Shooting)	2013	Bioshock Infinite	94	1518
		Battlefield 4	81	1490
		Crysis 3	76	721
Strategy	2015	Starcraft II: Legacy of the Void	88	1997
		Heroes of the Storm	86	1177
		Galactic Civilizations III	80	920
Strategy	2014	Endless Legend	82	944
		Hearthstone: Heroes of Warcraft	80	832
		Age of Wonders III	88	857

In Table 5.2.1, the result of comparison between games is shown. It can be observed that the score provided by the system is able to form the same ranking with the one given by Metacritic.com. However, this pattern is only applicable whenever the genre and year released of games is the same. The scores for strategy games in 2014 are in discrepancy with scores from Metacritic.com. This is mainly due to the game "Age of Wonders III", which is given a high score by Metacritic.com but not the system. Such discrepancies are sometimes inevitable. However, in such cases, the user can use the web visualization to analyze each aspect of the game in depth, in order to obtain better insights of what is actually happening.

CHAPTER 6: CONCLUSION

The video gaming industry has evolved to become a multi-billion-dollar global industry. There are plenty of game producers and publishers prepared to develop or release their game to the video games market. The competition between video game companies is a lot tougher than before. Therefore, a solution must be developed for video games companies to stay in the fierce competition. Recognizing trend is one of the ways that will allow game companies to make better decisions in produce games that are able to lure or attract video gamers. Trends can be recognized by analyzing the video gamer. There is an abundance of games review and user discussions on the internet that freely available. Games aspect ranking can be used to mine such large data on the internet.

In this project, a games aspect analysis system has been developed to help the company to analyze the online data and recognize trend.

For future work, the part of the system which involves game score prediction can be further enhanced to improve the results. Since the current system only uses the generated aspect ranking list to identify the score of each game, a future system may include other features such as review dates, game price and so on. In addition, it is also possible to train a machine learning model to help predict the scores by combining the aspect ranking list and other information to fit the scores in Metacritic.com.

Lastly, the web interface can be improved by turning it into a complete subscription system which allows user to register and login to the system by paying a certain amount of fee in order to visualize and analyze the result.

BIBLIOGRAPHY

Baccianella, S., Esuli, A., & Sebastiani, F. (2010, May). SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. In LREC (Vol. 10, pp. 2200-2204).

Bancken, W., Alfarone, D., & Davis, J. (2014). Automatically detecting and rating product aspects from textual customer reviews. In (To appear in) Proceedings of DMNLP Workshop at ECML/PKDD.

Brouwer, B. (2015). Twitch Claims 43% Of Revenue from $3.8 Billion Gaming Content Industry [online] Available at: http://www.tubefilter.com/2015/07/10/twitch-global-gaming-content-revenue-3-billion/ [Accessed 10 Aug. 2015].

Chen, D., & Manning, C. D. (2014, October). A Fast and Accurate Dependency Parser using Neural Networks. In EMNLP (pp. 740-750).

Comp.leeds.ac.uk, (2015). The University of Pennsylvania (Penn) Treebank Tag-set. [online] Available at: http://www.comp.leeds.ac.uk/amalgam/tagsets/upenn.html [Accessed 20 Nov. 2015].

Esldesk.com, (2015). List of Pronouns. [online] Available at: http://www.esldesk.com/vocabulary/pronouns [Accessed 20 Nov. 2015].

Hu, M., & Liu, B. (2004, August). Mining and summarizing customer reviews. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 168-177). ACM.

Hu, N., Pavlou, P. A., & Zhang, J. (2006, June). Can online reviews reveal a product's true quality?: empirical findings and analytical modeling of online word-of-mouth communication. In Proceedings of the 7th ACM conference on Electronic commerce (pp. 324-330). ACM.

Huang, Y. F., & Lin, H. (2013, April). Web product ranking using opinion mining. In Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on (pp. 184-190). IEEE.

Kleinberg, J. (1999). Authoritative sources in a hyperlinked environment. Journal of the ACM, 46(5), pp.604-632.

Kong, R., Wang, Y., Xin, W., Yang, T., Hu, J., & Chen, Z. (2011, October). Customer reviews for individual product feature-based ranking. In Instrumentation, Measurement, Computer, Communication and Control, 2011 First International Conference on (pp. 449-453). IEEE.

Li, S. K., Guan, Z., Tang, L. Y., & Chen, Z. (2012). Exploiting consumer reviews for product feature ranking. Journal of Computer Science and Technology, 27(3), 635-649.

Miao, Q., Li, Q., & Dai, R. (2009). AMAZING: A sentiment mining and retrieval system. Expert Systems with Applications, 36(3), 7192-7198.

McGlohon, M., Glance, N. S., & Reiter, Z. (2010, May). Star Quality: Aggregating Reviews to Rank Products and Merchants. In ICWSM.

Qiu, G., Liu, B., Bu, J., & Chen, C. (2009, July). Expanding Domain Sentiment Lexicon through Double Propagation. In IJCAI (Vol. 9, pp. 1199-1204).

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013, October). Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the conference on empirical methods in natural language processing (EMNLP) (Vol. 1631, p. 1642).

Talkenglish.com, (2015). Top 1500 Nouns used in English Vocabulary Words for Speaking. [online] Available at: http://www.talkenglish.com/vocabulary/top-1500-nouns.aspx [Accessed 20 Nov. 2015].

Universaldependencies.github.io, (2015). Syntax. [online] Available at: https://universaldependencies.github.io/docs/u/overview/syntax.html [Accessed 20 Nov. 2015].

Wu, J., Xu, B., & Li, S. (2011, July). An unsupervised approach to rank product reviews. In Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on (Vol. 3, pp. 1769-1772). IEEE.

Xiang, B., & Zhou, L. (2014). Improving Twitter Sentiment Analysis with Topic-Based Mixture Modeling and Semi-Supervised Training. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers) (pp. 434-439).

Xu, L., Liu, K., Lai, S., Chen, Y., & Zhao, J. (2013). Mining Opinion Words and Opinion Targets in a Two-Stage Framework. In ACL (1) (pp. 1764-1773).

Yu, J., Zha, Z. J., Wang, M., & Chua, T. S. (2011, June). Aspect ranking: identifying important product aspects from online consumer reviews. In Proceedings of the 49th

Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1 (pp. 1496-1505). Association for Computational Linguistics.

Zha, Z. J., Yu, J., Tang, J., Wang, M., & Chua, T. S. (2014). Product aspect ranking and its applications. Knowledge and Data Engineering, IEEE Transactions on, 26(5), 1211-1224.

Zhang, K., Cheng, Y., Liao, W. K., & Choudhary, A. (2011, August). Mining millions of reviews: a technique to rank products based on importance of reviews. In Proceedings of the 13th International Conference on Electronic Commerce (p. 12). ACM.

Zhang, L., Liu, B., Lim, S. H., & O'Brien-Strain, E. (2010, August). Extracting and ranking product features in opinion documents. In Proceedings of the 23rd International Conference on Computational Linguistics: Posters (pp. 1462-1470). Association for Computational Linguistics.

Zhuang, L., Jing, F., & Zhu, X. Y. (2006, November). Movie review mining and summarization. In Proceedings of the 15th ACM international conference on Information and knowledge management (pp. 43-50). ACM.