Analysis of advancing strategic game AI in the StarCraft II research environment

Yinheng Chen

A report submitted for the course
COMP4560
Supervised by: Penny Kyburz
The Australian National University

October 2019
© Yinheng Chen 2019
Except where otherwise indicated, this report is my own original work.

Yinheng Chen
25 October 2019
Acknowledgments

My thesis would not have been without your help, Thank you very much.

To Penny Kyburz, my project supervisor, the success and final result of the project would not have been without your continuous support. I would like to express my very great appreciation to you for your generous help and guidelines.

To all of my friends and family, thank you for your accompany and supporting me in my undergraduate.
Abstract

Human believe that they can benefit from AI research in games. Up to now, human have made a huge progress in the study of Go AI (AlphaGo). However, compare to real world, Go is more static and it lacks of diversity. Therefore, human is now trying to solve more complicated and dynamic games (RTS games). In this thesis, an simple introduction of real-time strategy (RTS) games is firstly presented. Specifically, we will focus on one of the most successful RTS game - StarCraft II, which has been thought as a new grand challenge for deep learning. Then, this thesis presents an overview of existing work on AI for StarCraft II. Among all, one of the famous and successful instance of StarCraft II AI is AlphaStar which is designed by Deepmind. In this thesis, we figure out which features are important to analyze AI. To prove these features are important, we build a classifier based on these features to predict AI's level. An example of analysis of AlphaStar is presented as well. By using our classifier, AlphaStar is categorized as a top StarCraft II player. Finally, We describe some problems that has been addressed by existing AI and some problems remain open.
Contents

Acknowledgments iii

Abstract v

1 Introduction 1
  1.1 Problem Statement ........................................... 1
  1.2 Motivations .................................................. 2
  1.3 Project Scope ............................................... 2
  1.4 Report Outline .............................................. 3

2 Real-time Strategy Games 5
  2.1 Real-Time Strategy Games ..................................... 5
  2.2 StarCraft II .................................................. 6
  2.3 Challenge in RTS Games (StarCraft II) ...................... 7

3 Related Work 11
  3.1 Existing work on RTS Games AI (StarCraft II) ............. 11
  3.2 Analysis of Players Skill .................................... 13
  3.3 Summary .................................................... 13

4 Design and Implementation 15
  4.1 OverView of Global Players in StarCraft II ................. 15
  4.2 Features to evaluate Players/AIs ............................ 16
    4.2.1 APM .................................................. 16
    4.2.2 SQ .................................................. 17
    4.2.3 Other Features ....................................... 17
  4.3 Modelling to extract features ............................... 19
  4.4 Modelling to predict Players/AIs league ................... 20
    4.4.1 Data Collection ....................................... 20
    4.4.2 Classification ......................................... 20
  4.5 Summary .................................................... 21

5 Results and Evaluation 23
  5.1 Analysis of AlphaStar ....................................... 23
    5.1.1 Micro-management ..................................... 23
    5.1.2 Macro Management ..................................... 24
    5.1.3 Summary ............................................... 28
  5.2 Performance of Classifier ................................... 28
Contents

6 Conclusion 31
   6.1 Future Work ........................................ 32

Bibliography 33

Appendix 1 - Project Description 35
   1 Description ........................................ 35
   2 Learning Objectives ................................ 35

Appendix 2 - Study Contract 37

Appendix 3 - Description of Artefact 39
   1 Model to decode StarCraft II replays .............. 39
   2 Model to predict players/AI’s league .............. 40
   3 Others .............................................. 41
   4 Testing .............................................. 41
   5 Acknowledgement .................................... 41

Appendix 4 - README 43
   1 Description ........................................ 43
   2 Requirement ....................................... 43
   3 Models ............................................. 43
      3.1 Model to decode StarCraft II replays ......... 43
      3.2 Model to predict players/AI’s league ......... 44
      3.3 Others .......................................... 45
   4 Acknowledgement .................................... 46
List of Figures

4.1 Player League Number - Location ............... 15
4.2 Player League Apm .................................. 16
4.3 Player League SQ ................................. 17
4.4 Attribute matrix ................................. 18
5.1 APM distribution ................................. 24
5.2 Economic features for AlphaStar and Mana ......................... 25
5.3 Units and Workers for AlphaStar and Mana ......................... 26
5.4 Supply Used/Made for AlphaStar and Mana ......................... 27
5.5 Spent resource distribution ......................... 27
List of Tables

3.1 Results of AI research on StarCraft II by implementing Reinforcement Learning .......................................................... 12

4.1 Avg Players MMR - League ....................................................... 18

5.1 Prediction of League for All ..................................................... 28
5.2 Prediction of League for Terran ............................................... 28
5.3 Prediction of League for Protoss ............................................. 28
5.4 Prediction of League for Zerg ................................................. 28
Chapter 1

Introduction

1.1 Problem Statement

The field of AI research for strategy games can be back to 1968 when an American Albert L. Zobrist designed his first Go program, even that program could only finish a match. From then on, AI research for strategy games has become an important part in computer science.

For decades, people have been trying to design an AI which can defeat professional players in strategy games. In 1996, a chess-playing AI Deep Blue won its first game against a world champion, which has been though as a milestone in AI research for strategy games.

The success of Deep Blue inspires human to explore a deeper level of strategy games. Computer scientists begin to do AI research on Go. Compared to chess, Go contains a much wider searching space ($10^{170}$). In the early 21st century, a number of Go AI which implemented Monte Carlo tree search, Minimax tree search or others appeared and defeated a great number of amateur Go players in next few years. However, the progress of Go AI stuck until 2015.

While people become pessimistic to AI research on Go, machine Learning and deep Learning become people’s hope. Nevertheless, the appearance of AlphaGo [Silver et al., 2016] tells people that AI has completely mastered Go. AlphaGo is designed by Deepmind. In 2015, AlphaGo defeated professional Go player Fan Hui with 5 : 0. In March 2016, AlphaGo defeated ‘legendary’ Go player Lee Sedol with 4 : 1. After that, it is more difficult for a master Go player to win only a match in the competition with AlphaGo.

AI research has made a huge progress on some strategy games, more specifically, some static strategy games. The ultimate target of AI research on strategy games is implementing these AI technology on real-life. Therefore, a great number of AI research on RTS Video games are doing in the World.

Among all of RTS Video games, StarCraft and StarCraft II have been thought as
a very good example to implementing AI research. StarCraft was released on 1998 and previous AI research on StarCraft has made some progress. Since StarCraft II released in 2010, the AI research on StarCraft stuck. StarCraft II inherits some elements from StarCraft but they are still totally different. There are some reasons why many people stop StarCraft AI research and turn to StarCraft II AI research, even some people have made a progress on StarCraft AI research, this will be discussed in Chapter 2.

In the current world, AI research on StarCraft II has advanced significantly. The most famous StarCraft II AI bot is AlphaStar [Vinyals et al., 2019] which is designed by Deepmind as well and appeared in 2018. AlphaStar defeated two StarCraft II professional players in December 2018. However, only two StarCraft II professional players were defeated until now. Also, some people do not recognize AlphaStar because these two master players are not top professional players and AlphaStar takes some unfair advantages at that moment. People have doubted that will AI master RTS games in the future?

1.2 Motivations

The field of AI research in RTS games was started years ago when Go AI was struggling to defeat amateur players. The first idea of RTS games can be viewed as a new AI challenge was posted in [Buro, 2003] where author believes that RTS games are ideal test applications for real-time AI research. Since Michael Buro's call for AI research in RTS games [Buro, 2004], human has made a progress in the field of AI research for RTS games. In [Ontanón et al., 2013], it provides a overview AI research on StarCraft. However, StarCraft II and StarCraft are totally different video games and the AI research on StarCraft has stucked. Although there exist a great number of AI researches on StarCraft II, a overview of existing work is lacked. In that case, I think a overview of existing work of AI research on StarCraft II will benefit more people who are interested in this field.

Compared to traditional strategy games, it is a narrow scope to judge a AI by only whether it win or lose in competition since RTS games contains much more elements. Also, there lacks efficient approaches and criterion to evaluate StarCraft II AI.

So, my intention in this project is providing a overview of existing AI research on StarCraft II and figure out a approach to analyze StarCraft II AIs, I think this would help people to improve and analyze their StarCraft II AI.

1.3 Project Scope

Generally, people judge a Game AI by its winning rate in a series of competitions/matches. In my project, I design a approach to analyze StarCraft II AIs based on some criterions which are tested in real PvP(PLAYER vs PLAYER) matches. Also, my project
provides a overview of existing work on StarCraft II AI research and explain their advantages and weakness.

1.4 Report Outline

In Chapter 2, I will explain what RTS games are, more specifically, StarCraft II and why it become a challenge for AI research. In Chapter 3, some existing work on StarCraft II AI research and one existing approaches to analyze StarCraft II AI are presented. In Chapter 4, I will discuss which tools I used and how I build my model to analyze AI. In Chapter 5, I will focus on analysis on AlphaStar, explaining the difference AlphaStar with professional players in Strategic level and their advantages and weakness. Also, the results of my approach to evaluate AI is demonstrated in this chapter as well. Chapter 6 is my conclusion of my project.
Real-time Strategy Games

2.1 Real-Time Strategy Games

A strategy game is a game in which players can control multiple units and it requires players to have a ability to make decision. Players’ decisions usually have a crucial impact on the outcome of the game. To be the winner of the game, players usually need to defeat their opponent by achieving some targets such as eliminating units of opponent, occupying opponent’s base and etc. The strategy games can be generally categorized into two subtype based on how it progress: turn-based and real-time. Game Chess, Go and Chinese Chess are most common turn-based strategy games in the World. A strategy Video game usually has more units and larger action space than traditional strategy game. Total War, WarCraft, StarCraft and Command & Conquer: Red Alert are top popular strategy Video games.

Real-Time strategy (RTS) games is a kind of strategy Video games in which the games does not progress in turns. A match of a RTS game usually happens on a specific map. Usually, a standard RTS game contains three main components:

- Training military power: The most important component for a player to achieve winning goal in most RTS game. Players are required to train and control their army to beta the army of their opponent and destroy opponents’ base to win the game. Military units usually are trained in specific buildings.

- Collecting Resources and Resources Management: In most RTS games, resources can be used to construct buildings, train units and develope technology. Players use their workers to collect resources.

- Technology development: Players are allowed to do research on technology to improve the quality of their units and increase their resources collection rate. Also, a technology can be prerequisites of some units or other technology.

Also, a RTS game has other components such as map (size of the map, terrain in the map), match type (1v1, 2v2, NvN or Melee), tactics and etc.

The other charm of RTS games is that players can use their unique operations to
give their units/buildings instructions to achieve some goals, which means even a player has some disadvantages in above three components in a match, he/she can still defeat their opponent if he/she performs good operations on his/her units.

Based on above features, a RTS game can be seen as a simplification of a instance in real-life environment.

According to [Buro, 2004], arguments support AI research in RTS games are:

- Experiments can be conducted in well-defined game environments and RTS games provide some built-in functions to measure performance
- RTS games can be decomposed to smaller aspects such as win a local battle, constructing base or army buildings, etc.,
- Currently, for some PvE games (Player vs AI/bot), players would be tired to the games soon once they find the law of bots. More Intelligent games built-in bots provides a better game experience for players.
- Helping to improve the quality of games. For example, alleviating human players from repeated manually tasks, unit balancing, finding potential games bug, etc.,

### 2.2 StarCraft II

StarCraft II, whose predecessor is StarCraft: Brood War, was firstly released by Blizzard Entertainment in 2010 and still actives in the video game markets. StarCraft II is thought as one of the largest and most successful, one of the most complex dynamic, one of the longest-played esports RTS video games in the world. Compared to some traditional RTS games, StarCraft II has following extra features:

- Although StarCraft II is firstly released in 2010, its’ picture quality is still much better than RTS games, even its picture quality is on top level among all video games.
- StarCraft II’s predecessor: StarCraft is the milestone of RTS games, StarCraft II inherits many advantages from its predecessor:
  - An attractive background and game story. Players can play some specific roles and enjoy the game story in game campaign.
  - A perfect economy systems: There are two types of resources: Mineral and Vespene. These resources are limited and separate on the different areas of the map. Workers are allowed to collect different resources from ground and deliver resources to the closest base. Constructing buildings, researching technology, training units costs mineral and/or vespene.
  - A Supply System: Compared to other RTS games, training units also requires enough supply in StarCraft II, players need cost mineral to produce supply.
Diversity of units: In StarCraft (II), players must choose one of three races, Terran (Human), Zerg (merciless beasts) and Protoss (an ancient and noble race which has advanced psionic technology) [Blizzard Entertainment, 2019]. Different races have different mechanism, units and buildings. The difference among races are significant. Also, units can be divided into two groups: ground units and air units.

Besides, StarCraft II have totally new macro mechanism Compared to StarCraft. For example, much humanize operations (players are allowed to click 96 units in a group in StarCraft II while a group only has at most 12 units in StarCraft II), more maps, more units, wider action space and etc.

- There are frequent patches and updates on StarCraft II, which means the marco mechanism, features of units and buildings, maps database for competition are frequently changed while StarCraft has been keep stable for more than 10 years.
- A awesome editor, which allowed players to create their own buildings, unique units, resources ground add events and etc. in their own maps.
- Blizzard Entertainment and its partners Deepmind, MIT and others provide strong support for AI research on StarCraft II. Open tools and API (pysc2, s2protocol, s2client and etc.) can be accessed for everyone.
- Currently, various of StarCraft II AI research communities exist online. Also, there are some organizations hold StarCraft II AI competition in which people upload their AI and compete with other players/AI. StarCraft 2 AI Ladder ¹ is a very famous one.

2.3 Challenge in RTS Games (StarCraft II)

In [Buro, 2004] and [Ontanón et al., 2013], both they define 6 challenges for RTS games. While RTS games become more complicated, the challenges become more harder. Generally, we have following challenges in RTS games (StarCraft II):

1. **Planning**: In RTS games, matches are executing maps, a map in RTS games can be thought as checkerboard to chess/Go. Compared to constant size of 19 x 19 checkerboard of Go. The maps size of StarCraft II can be range from 100 x 100 to 256 x 256, which means the state space in RTS games is immense. Even there are approximately $10^8$ possibilities for just selects actions among a combinatorial space, [Vinyals et al., 2017]. Additionally, number of actions and units are variable and actions happen simultaneously. Multiple game levels are executing simultaneously (local battle, training units, economy constructing, etc.).

2. **Learning**: Recently, the resurgence of deep learning by using neural networks yielding significantly success in the game of Go [Vinyals et al., 2017].

¹https://sc2ai.net/RecentMatches.php
extremely large domain, multi-agent problem, match of different races, even pose a new grand challenge for reinforcement learning.

3. **Imperfect information**: In StarCraft II, each unit can detect a small range of area. Unlike each player can observe each other’s action in Go, players are only allowed to observe a partial map based on the location of their units, more specifically, there are three situations:

(a) Player can have full view of an area if he/she currently has units on the area.

(b) Player can have a last view of an area if he/she has units exploited the area before, however, players do not know what is happening in the area now.

(c) A area is invisible for a player if he/she has no unit scout the area before.

Also, since the map is large, it is extremely difficult for players to catch all events which simultaneously happen on those areas that they have view. Therefore, AI may need build extra model to predict what opponent is doing. Therefore, players should regularly send scouts to localize and spy opponents' bases and army.

Another aspect is that players are allowed to choose random race among three races, which means the opponent would not know the race of the player until the opponent first observe the player during the match.

4. **Spatial and Temporal Reasoning**: In the current competition of StarCraft II, various strategies are used by professional players, each strategy has its own timing process. For example, a player usually constructs his/her second base between 1:30 - 2:30 game time if he/she takes expansion strategy, otherwise, the player is probably take rush strategy. A rush strategy is that a player do not develope his/her economy and spends more resources on training army and prepare to attack the opponent soon, while the opponent choose develope economy first.

Different units has different attack range and a unit is allowed to attack another only if the unit has the vision on another. Also, units on high ground has an advantage than units on low ground since units on low ground lose vision onto the high ground unless the unit is air unit. Additionally, some units are only allowed to attack ground and some units only attack air units.

5. **Domain Knowledge Exploitation**: In most static strategy games, game rules are unchanged, the relations among units are certain. So, computer scientists can exploit the vast existing domain knowledge to create good evaluation functions to accelerate algorithm [Ontanón et al., 2013]. However, there is no clear relationship among units and no clear rules to restrict actions of units. No clear domain knowledge exists in RTS games. Therefore, it is very difficult
for researchers to design good heuristic functions and evaluation functions to improve search and exploitation.

6. Complicated game Level: There exists various game levels in StarCraft II. A great number of events with different types are executing in the game. Typicall, we can divided events as follows:

- Macro level: Where players need to find a efficient way to spend their resources and balancing resources distribution to maximize the number and the quality of their army power, this includes;
  - Army training: when begin train units? How many army buildings is enough to train units ?
  - Economy development: when to expand economy such as constructing another rbase? How many workers are most efficient or proper?
  - Technology research: Technology costs resources but improves quality of units. Is it necessary to research technology? More units maybe more powerful than higher h quality units.
  - Supply: Units require supply. Fewer supply restrict the training of units while higher supply waste resources.

- Micro level: Players are able to control units simultaneously and individually by giving different instructions such as move, attack, stop, patrol, collect resource, hit-and-run (keep our units’ distance to opponents’ units when our units can attack enemy while enemy cannot attack ours due to a shorter attack range). Many experiments have proven that AI has ability to do much better in this level because one advantage of bots is that bots can simultaneously process multiple micro tasks and show accuracy results.

7. Game update: Unlike the law of chess and Go has not changed for decades. To balance units and keep players interest in StarCraft II, Blizzard frequently updates StarCraft II per 2 to 3 months. For each update, various of new maps release, features or costs of units/technology/buildings change, units/technology/buildings adding or remove, even the initial state of game can be modified. The macro mechanism of three races may be changed. Some strategies are efficient in current version may be discarded soon once new patch released. Whether people can design AI is robust to the regular update of StarCraft II is still an open question.
Chapter 3

Related Work

3.1 Existing work on RTS Games AI (StarCraft II)

RTS games have advanced significantly for 20 years. However, hard-coded approach is still first choice for developers to design built-in AI for commercial RTS games. Advantages of hard-coded approach include easy to implementation and modification, multiple difficult levels of AI and more. For many existing work by taking advanced approaches to build AI, they still begin test their AI by playing matches with built-in AI. As we all know, hard-coded approaches are struggling on dynamic environments, opponents are easy to find characteristics of hard-coded approaches and use some simple targeted strategies to defeat hard-coded based AI.

Although there are multiple game levels in RTS games, various methods show their adaptivity for most game levels, which means we can use same method to build micro or macro management in RTS games. Previous work on RTS game AI usually use planning, tree-search or machine learning. A notable example of planning approach is real-time case-based planning [Ontañón et al., 2008] which shows adaptivity of AI on dynamic environment in classical RTS games. Goal-driven autonomy based AI [Weber et al., 2010] has a overall 73 % winning rate to built-in AI in StarCraft. Summary of classical approaches focus on difficult aspects of RTS games can be seen in [Ontanón et al., 2013; Robertson and Watson, 2014].

Recently, with the significant development of deep learning, genetic algorithm based AI has made dramatically improvement. For specific game level in StarCraft II such as build order (macro management), a multi-objective genetic algorithm can be used to find the optimization [Köstler and Gmeiner, 2013; Kuchem et al., 2013]. Similarly, multi-objective genetic algorithm presents a advantage on units control (micro management) by optimizing the behaviors of units [Schmitt and Köstler, 2016].

Nevertheless, some of above approaches based AI are struggling to defeat insane-level built-in AI (insane-level AI has extra resources in each match) while the rest only reach under average players’ level. The breakthrough of AI research for strategy games is the rise of deep reinforcement learning. The techniques of reinforcement learning achieves in success in Atari games and result in the first agent which has the
capability of being expert in various challenging tasks [Mnih et al., 2015]

Another successful example by using reinforcement learning combined with tree search to master the strategy game is AlphaGo, which achieved a 99.8 % winning rate against other Go AI [Silver et al., 2016].

Although reinforcement learning has shown great success in classical strategy games, whether it can be used to master StarCraft II is still unknown. However, some progress has made. The first deep reinforcement learning based AI made a tie result against the easiest built-in AI [Vinyals et al., 2017]. Following table shows the results of some existing work by implementing reinforcement learning combined with another approach against StarCraft II built-in AI.

<table>
<thead>
<tr>
<th>Author</th>
<th>approach</th>
<th>Built-in AI level</th>
<th>Winning rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Lee et al., 2018]</td>
<td>Modular Architecture</td>
<td>Level-5</td>
<td>92%</td>
</tr>
<tr>
<td>[Shao et al., 2018]</td>
<td>Curriculum Transfer Learning</td>
<td>Local battles</td>
<td>100%</td>
</tr>
<tr>
<td>[Sun et al., 2018]</td>
<td>Flat Action Structure</td>
<td>Level-10</td>
<td>81%</td>
</tr>
<tr>
<td>[Pang et al., 2019]</td>
<td>Hierarchical approach</td>
<td>Level-7</td>
<td>93%</td>
</tr>
</tbody>
</table>

However, AlphaStar has made a dramatic progress since 2017, its predecessor which is based on canonical deep reinforcement learning can only make a tie result against the easiest built-in AI [Vinyals et al., 2017], it defeated two master players in December 2018. AlphaStar is capable for playing the full game of StarCraft II by using a deep neural network that is trained directly from raw game data by supervised learning and reinforcement learning, it also implements a novel multi-agent learning algorithm [Vinyals et al., 2019]. In [Arulkumaran et al., 2019], it exposed some algorithms that are used to train AlphaStar, such as Lamackian evolution, competiite co-evolution and quality diversity.

Recently, Blizzard and Deepmind has applied AlphaStar in rank match of StarCraft II. Every player can play against AlphaStar once their application is successful. However, the performance of AlphaStar in rank match is undisclosed. Additionally, the matches between two professional players and AlphaStar lack more variables. For example, all matches are 1 v 1 and the race of AlphaStar and professional players are both Protoss. In Rank match and competition of StarCraft II, risky strategies are wildly used, but these two professional players choose to use stable strategies in all matches. Although, AlphaStar has been thought most successful StarCraft II AI, whether it can master StarCraft II or not is still unknown.

In StarCraft 2 AI Ladder 1, various of machine learning based AI are also open

---

1https://sc2ai.net/RecentMatches.php
for everyone. Also, this website provide the results of competition among AIs. However, the performance of these AIs play against human players is still difficult to evaluate.

3.2 Analysis of Players Skill

For those AI researchers, one problem is that they cannot execute their program on ranked match. Playing against built-in AI has no answer of what human players level of their AI could be as a novice player can easily defeat level-10 built-in AI by some tricks. Also, some criterion may help AI researchers to improve their program.

Few work about analysis of players skill can be found online. A previous work which may help to evaluate the ability of AI can be seen in [Avontuur et al., 2013] which used a SMO classifier to construct a player model to predict player skill. In ranked match, players are divided into 7 division based on their performance in hundreds of matches. This model achieved a weighted accuracy of 44.9 %.

3.3 Summary

In this chapter, some existing work of StarCraft II AI research is presented. Currently, machine learning based method and reinforcement learning are still predominant on AI research for StarCraft II. Also, a previous work about modelling players skill is presented, however, the accuracy of this model is not satisfied.

In next chapter, an overview of players of StarCraft II will be first presented. Then, we will find out features which are important to evaluate a player/AI. A model which used those features to predict AI/player division will also be introduced.
Related Work
4.1 OverView of Global Players in StarCraft II

To find out which features are related to analysis Game AI. We first need to understand the overview of global players and how human players are evaluated in Ranked games. Following picture shows Player League (division) distribution in different servers. The data is retrieved from rankedftw\(^1\).

![Figure 4.1: Player League Number - Location](Image)

In StarCraft II ranked games, human players are divided into 7 leagues: Bronze, Silver, Gold, Platinum, Diamond, Master, GrandMaster with ascent order based on their performance in a number of matches. There are 300,000 active players in StarCraft II ranked games. As we can see, most players’ leagues are in the range of Silver to Diamond. Generally, we think the level of human players who are in Gold league is equals to that of Level-10 built-in AI, which means the performance of most

\(^1\)https://www.rankedftw.com
reinforcement learning based AI is under average performance of human players.

In each server, there exist at most 200 GrandMaster players. The total number of GrandMaster players in the World is no more than 800. So, only approximately 0.27% players can be GrandMaster in ranked matches.

4.2 Features to evaluate Players/AIs

Although the result of a match is clear, features which impact the result cannot be neglected. For example, how a player control his/her units in a local battle and does he/she control his units in time? Does he/she has a good solution to balance supply/resource/army? In StarCraft II, the most two important features are APM (action per minutes) and SQ (Spending quotient).

4.2.1 APM

In RTS games, players give instructions to units by clicking mouse or typing keyboard. These clicking actions can usually be divided two macro actions and micro actions. The computation of Average APM is

\[
\text{Average APM} = \frac{\text{Total Num. Of Actions Events}}{\text{Game length (in minutes)}}
\]

A higher APM of a player means that the player has the ability to process more events in each unit of time in the game. Although, a player has high APM does not mean he/she is a advanced player, high-level players usually should have a higher APM. The following image shows the average apm of global players in each league of ranked match in StarCraft II. As we can see, with the ascent order of league, the average APM increase. While the average APM of GrandMaster is approximately 300 (300 actions per minutes), those players in Bronze league can only give one instruction per second (60 APM).

![Figure 4.2: Player League Apm](image)
4.2.2 SQ

While APM more focuses on micro management, e.g., High APM players take advantages in local battle, macro management such as building high-performance economy system is also important. Players collect resource and spend resource on numbers of aspects. To maximize the use of resource and to build a more efficient economy system, players should keep less resource unspent and increase resource collection rate by expanding and training more workers. SQ is a feature to quantitatively measure players’ economic management in a game and its calculation is shown below [Team Liquid, 2019]

\[ SQ(i, u) = 35(0.00137i - \ln(u)) + 240 \]

where \(i\) represents resource collection rate and \(u\) represents unspent resource, the resource can be calculated as weighted arithmetic mean of Mineral and Vespene, here the weight of Mineral and Vespene ranges from 1 : 2 to 1 : 3. Unlike the measurement of APM, players who have higher should SQ be considered as advanced players. Following image shows the average SQ of global players in each league of ranked match in StarCraft II. The gap of average SQ among leagues is not significantly as that of APM, however, players in higher leagues exactly have higher SQ.

![Player League SQ](image)

**Figure 4.3: Player League SQ**

4.2.3 Other Features

Figure 4.4 shows attributes matrix of features in 7 leagues, the value of these features are averaged of recent 300,000 matches, the original statistics is retrieved from \(^2\). Some features should be understood to evaluate players/AIs’ level. The direct feature to reflect players’ level is Players MMR, other features which strongly positively or negatively related to Players’ MMR should be considered.

1. **Avg Players MMR**: In StarCraft II, MMR is players’ real score in ranked matches, players will earn MMR if he/she win a match and vice versa. Once a player’s

\(^2\)https://sc2replaystats.com/stats/analytics
MMR reach a threshold, he/she will be upgraded to next league. The Avg Players MMR in each league is shown in table 4.1

2. **Avg Game Length**: which is the average game length in 7 leagues, since StarCraft II is fast paced game, high-level match usually has shorter game length

3. **Avg Workers**: Workers can collect and deliver resource, a player should train more workers when he/she tries to expand.
4. **Avg (Resource) Collection Rate**

5. **Avg Unspent Resource**

6. **Avg Units Produced**: How many units are trained by a player in a match.

7. **Avg Units Killed**: Players win local battle by eliminate opponents units. Excluding the characteristic of different races, the winner usually killed more units than the loser.

8. **Avg Workers Killed**: To win a match, player can also effect opponents’ economy which leads to opponents train less army. The economic strategy also includes restricting opponents economy, therefore, killing opponents workers will strongly damage opponents’ economy.

9. **Avg Supply blocked**: Since training units require supply, players are not allowed to train units if he/she lacks of supply. A good player should construct enough supply in advance to avoid supply blocking training of units.

10. **Avg Supply Used/Made**: This feature is intuitive, lower avg supply Used/Made means players waste resource in building too much supply or players lost many units in this moment. However, too large value of this feature means players are experiencing potential supply blocked.

### 4.3 Modelling to extract features

Features which related to players/AIs’ performance in match have been shown in previous section. The only way that we can access player’s performance is using .SC2Repaly file which represents a replay of a match in StarCraft II. This file can be used as a replay by executing StarCraft II game. However, watching hundreds and thousands of replays to record features in StarCraft II game program is costly. Therefore, an automatic method is required.

To decode Sc2Replay file, only two modules can be used. Sc2reader is a module which can decode Sc2play file to a Python class, however, this module is not adaptive to matches of new version and customized matches. S2protocol is also a module which designed by Blizzard, this module strongly adaptive to matches in all version StarCraft II because Blizzard would update this module once they update StarCraft II. In this module, all elements about the replay are only encoded into a great number of events. This module only provides events and does not answer any question. For example, if we want to calculate the APM of players, we need extract and select those events which related to players’ actions and calculate frequency of these events in a unit time. To use s2protocol to decode replay files, it requires people have deep understanding of StarCraft II. Even Sc2reader has some useful functions, by considering the adaptivity of it, I finally used S2protocol as a bottom and build top on it.
4.4 Modelling to predict Players/AIs league

In early sections, we have described which features are important to analysis players'/AIs skill. However, it is costly for a person to understand all features if he/she does not have a long game experience in StarCraft II. Additionally, one intuitive feature to determine Players/AIs is league in ranked match. Since AIs are not allowed to play ranked match, we can build a classifier based on extracted features from human players’ replay to predict AI’s league.

4.4.1 Data Collection

Since there is no existed data, I collected replay files of recent 3 months from multiple websites. The total number of replays I have collected is 4917. The replays were originally uploaded by global players who posted their own games. I only extracted and store full-game data from 1 vs 1 ranked games. Therefore, each replay provided information about two players.

After filtering some replays and instances (eg. those replays have total game length less than 2 minutes, players have no rank), I created a dataset of 4114 instances. The instances were distributed over the league as follows: Bronze 14, Silver 128, Gold 500, Platinum 1355, Diamond 1685, Master 357, GrandMaster 244. As experienced players are more likely to be involved in StarCraft II community and upload their games, it not exceeds our expectations that the dataset contains few instances of Bronze or Silver players.

As no one is intended to build a AI which only reaches Bronze or Silver level in StarCraft II league and the insufficiency of instances whose players are in Bronze and Silver ranked games. I removed all these instances and built a classifier based on players whose league is between Gold to GrandMaster. After filtering, the dataset remain instances of 3981.

4.4.2 Classification

Note that difference races have different mechanism. For example, units of Zerg usually is cheaper but low quality while Protoss units are more expensive. Zerg players usually train much more units than Protoss players, which also requires Zerg players have a higher APM. Also, the base of Zerg is cheaper, which means Zerg is easier to expand and Zerg players usually have a higher resource collection rate. By considering different mechanism of different races, we need to category the dataset baed on players’ race.

Thanks to the development of machine learning, a great of number of classifiers are existed. After pre-processing and categorizing data, I performed executions on

---

5 classifiers: Linear classifier, Logistic Regression, Naive Bayes, Random Forest and Gradient Boosting Classifier. I found that Gradient Boosting Classifier outperformed all the other classifiers.

4.5 Summary

In this chapter, I first presented an overview and statistics of global players. Based on the statistics, I determined which features are important and should be used to evaluate players/AI.

In the next section of this chapter, I first explained which file contain players’ features in matches. Based on the s2protocol which is provided by Blizzard, I designed my own model which can be used to extract features from replays.

Since AIs are not allowed to play ranked games. To predict AIs league, I collected a number of replays of human players and builded a classifier based on data of these replays. By extracting features from replay between AI and AI(human), we can use the classifier to predict AIs’ league.

In next chapter, I will present a analysis of AlphaStar, explain the difference between AlphaStar and professional players. Additionally, I will use the classifier to predict AlphaStar’s league in ranked games.
5.1 Analysis of AlphaStar

As mentioned above, AlphaStar played against two professional players (TeamLiquid Mana and TeamLiquid TLO) in December 2018. In early 2019, Deepmind released 11 replays of these matches, the replays can be downloaded from Deepmind website. I used my replay decode model to extract some interesting features from these matches. Until now, the performance and features of AlphaStar can only be accessed from these 11 replays. Both AlphaStar and professional players choose Protoss to play against each other. Since TLO’s first preference race is not Protoss, 5 matches between TLO and AlphaStar are discarded. Also, the exhibition game is discarded. Therefore, we focus on the matches between Mana and AlphaStar. In this chapter, the analysis is divided into two sub-configurations: micro-management and macro-management. Also, same race matches (Protoss vs Protoss) are easier to observe difference between human players and AIs since the difference among macro mechanisms for different races is noticeable. Also, I did not average the value of features because of following reasons:

- We only have 5 samples.
- Different matches executed in different maps, different maps have different size which may effect features
- Players choose different strategies in each match, which significantly cause the difference of features.
- The games lengths are significantly different.

5.1.1 Micro-management

Figure 5.1 shows the histogram of APM for AlphaStar and Mana in a single match. As we can see, AlphaStar has an average APM of around 260, significantly lower than that of around 376 of Mana. However, the actions of AlphaStar is exactly more precise than Mana since human players are easy to produce useless actions. Additionally,
we should consider that RTS players will frequently click mouse and keyboard to warm up their hands in the first few minutes of the game, although many clicks are redundant. As we can see from the figure, the histogram of APM for Mana follows normal distribution while the distribution of APM for AlphaStar is approximately cumulative. We may also notice that sometimes APM for AlphaStar reach 1200 (20 actions in a second) and most of its actions are precise. By watching replays in StarCraft II, AlphaStar would reach extremely high APM in local battle against Mana, while it is impossible for a human player to have 20 precise actions per second in battles. Because of the advantages of APM for AlphaStar, it is reasonable that people (see [Timothy B. Lee, 2019] and StarCraft II communities) doubt AlphaStar defeats human players in local battles by machine characteristics (superhuman speed) rather than strategy (intelligence).

5.1.2 Macro Management

1. **Economy**: Figure 5.2(a) shows the resource collection rate for Mana and AlphaStar in a match. Figure 5.2(b) shows the SQ of AlphaStar and Mana in the match and Average SQ for global players in different league in Protoss vs Protoss matches. As we can see, although AlphaStar takes slightly advantages than Mana in resource collection, their economic strategies are almost same, eg their resource collection rate are almost same in many timeslices. However, we can see that the SQ for AlphaStar is also above averaged SQ of GrandMaster. Although AlphaStar has no advantages than professional player in SQ, we think AlphaStar still excel to economic management.

We may notice some outliers in figure 5.2 of resource collection rate for AlphaStar. AlphaStar’s resource collection rate is extremely low in around time
250. This is because Mana harasses (using a small number of units to damage enemy) AlphaStar’s workers which prevents AlphaStar’s workers collecting resource. By watching 5 replays between AlphaStar and Mana, we find that human player usually have strong intention to harass opponents’ workers (economic system) while AlphaStar not.

(a) Resource collection for AlphaStar and Mana

(b) SQ for AlphaStar and Mana

Figure 5.2: Economic features for AlphaStar and Mana
2. **Supply and Units**: To defeat the opponent, building an efficient economic system is just a cornerstone. Players should train a powerful army based on their economy. Figure 5.3 shows the number of produced units and workers for AlphaStar and Mana in a match. In StarCraft II, the common knowledge is that the number of workers on a Mineral and Vespene ground is 16 + 6 = 22 (since training workers also costs resource), although many players know that the most efficient number of workers is 21 + 6 = 27. As we can see from figure 5.3(b), Mana stops training workers at time 120 until time 300 when he constructs the second base to expand, while AlphaStar keeps training workers from time 0 to time 400. Since AlphaStar pursues the most efficient number of workers, it has a higher resource collection rate, therefore, AlphaStar trains more units than Mana.

![Produced Units Number](image)

**Figure 5.3: Units for AlphaStar and Mana**

Figure 5.4 indicates supply used/made for AlphaStar and Mana. In the first 500 seconds of the game, the values of supply used/made of AlphaStar and Mana have no significant difference. AlphaStar loses supply management when it has a higher number of units, while Mana keeps supply used/made stable until he lost units because of failure in battles (the 700s). Additionally, both AlphaStar and Mana experience supply blocked (supply used/made = 1) in some time. Overall, professional players still have a better supply management than AlphaStar.
§5.1 Analysis of AlphaStar

3. **Utility of Resource**: Players should find a balance of resource distribution to train a high-quality and higher number of army units. Chart 5.5 demonstrates percentage of spent resource in army, economy and technology for AlphaStar and Mana in a match. We notice that Mana spent more resource on economy and technology research, while AlphaStar spent more resource on training army. This condition also happens in other 4 matches. We can conclude that AlphaStar prefers to train a army contain more units, while Mana was trying to find a balance between number and quality.

![Figure 5.4: Supply Used/Made for AlphaStar and Mana](image)

![Figure 5.5: Spent resource distribution](image)
5.1.3 Summary

AlphaStar learns from hundreds and thousands of human players’ replays [Vinyals et al., 2019]. Although AlphaStar has different understanding in some aspects of StarCraft II (e.g., number of workers on resource ground, utility of resource.), AlphaStar still has no significant difference with professional players in regards to macro management. Also, AlphaStar does not create a complete new strategy on demonstrations. The reason why AlphaStar can defeat professional players is mainly because AlphaStar superhuman micro management.

5.2 Performance of Classifier

I used the Gradient Boosting Classifier to build a player model and used those features (see section 4.2) from replays to create the dataset. The results of the classifier to predict league for different races are shown in below tables:

<table>
<thead>
<tr>
<th>Table 5.1: Prediction of League for All</th>
<th>Table 5.2: Prediction of League for Terran</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Accuracy 0.57</td>
<td>Average Accuracy 0.62</td>
</tr>
<tr>
<td>League</td>
<td>League</td>
</tr>
<tr>
<td>Gold</td>
<td>Gold</td>
</tr>
<tr>
<td>0.49</td>
<td>0.70</td>
</tr>
<tr>
<td>Platinum</td>
<td>Platinum</td>
</tr>
<tr>
<td>0.59</td>
<td>0.43</td>
</tr>
<tr>
<td>Diamond</td>
<td>Diamond</td>
</tr>
<tr>
<td>0.58</td>
<td>0.68</td>
</tr>
<tr>
<td>Master</td>
<td>Master</td>
</tr>
<tr>
<td>0.38</td>
<td>0.42</td>
</tr>
<tr>
<td>GrandMaster</td>
<td>GrandMaster</td>
</tr>
<tr>
<td>0.57</td>
<td>0.60</td>
</tr>
</tbody>
</table>

As we can see, the performance of the classifier improves after we categorizing races. We notice that result of classifying GrandMaster instances is relatively high than other instances even it has the fewest instances. I assume that this is because the top GrandMaster players are almost professional players, whose features in replays is outstanding and significantly different with normal players. Additionally, it is easy
to access replays of professional players, therefore, many GrandMaster instances are also professional instances.

Also note that the all cases are harder to classify Master players. I believe this is mainly because the Master instances are fewer. Another reason is that even Master is middle of Diamond and GrandMaster, because of rarity of GrandMaster, the difference of features between Master players and Diamond players is not as noticeable as that between Master players and GrandMaster players. Therefore, a Master instances is more likely to be misclassified as a Diamond instances.

As the performance of classifier is satisfied in GrandMaster instances and we also believe that AlphaStar should be categorized to GrandMaster league based on demonstrations, I extracted 5 AlphaStar instances in 5 matches between AlphaStar and Mana and used the classifier to predict the league of AlphaStar. Expectedly, all 5 AlphaStar instances are categorized as GrandMaster, which means AlphaStar does reach a GrandMaster (professional) level in StarCraft II.
Conclusion

In my project, I first presented an overview of recent work for AI research on RTS games (StarCraft II) and explained why RTS games should be called in AI research. This is because RTS games can simulate a real dynamic instance of the world and the AI research on RTS games can potentially be implemented in reality.

Because of immense searching space in StarCraft II, traditional methods such as hard-coded, planning and tree search are not sufficient to build a strong AI. Reinforcement learning seems to be an excellent method to solve the problem. A notable example is AlphaStar.

To determine the skill of AI, I designed a model which can be used to decode StarCraft II replay file. Besides, I used my model to extract features from 4917 StarCraft replays and construct them to a dataset. I used gradient boosting classifier to model players’ skills based on my collected data. My classifier has a more precise classification than previous work (see [Avontuur et al., 2013] whose model has an average accuracy of 44.9%). By both visual analysis and prediction of classifier, we can easily made a conclusion that AlphaStar reaches the GrandMaster players’ level.

We noticed that AlphaStar did not exceed professional in macro management whether AlphaStar can defeat top professional players without advantages of micro management is still unknown. To reality, whether reinforcement learning can help human design a complete new strategy is still open problem. However, we should be optimistic as we have witnessed the huge progress of AI research for StarCraft II from 2017.

Considering situations that the lack of overview on AI research for StarCraft II, insufficiency of evaluation method for AI skill on StarCraft II, and AIs are not allowed to play ranked matches, I hope my work and tools can help people to analyze and improve their AI as well as encouraging people who are interested in AI research for RTS games.


6.1 Future Work

Since the insufficiency of replay instances, more high quality matches are required to improve the classifier. Also, StarCraft II has 4 servers: US, KR, EU and CN. The average skill of players in different servers are different. For example, players are ranked to Diamond league in KR server might be Master leagues in other servers. Therefore, dividing dataset based on servers may improve the performance of the classifier.

Currently, we only have a few of sample replays of AlphaStar. Also, we still have no much news and details about AlphaStar since February 2019, I will keep focus on Deepmind once it release something new about AlphaStar.

Based on the work I have done, it is challengeable but interesting to build my own StarCraft II AI in the future. I believe that I may find more interesting features when I am building my own StarCraft II AI.
Bibliography


AVONTUUR, T.; SPRONCK, P.; and VAN ZAANEN, M., 2013. Player skill modeling in starcraft ii. In Ninth Artificial Intelligence and Interactive Digital Entertainment Conference. (cited on pages 13 and 31)


starcraft. *IEEE Transactions on Computational Intelligence and AI in Games*, 5, 4 (2013), 293–311. (cited on pages 2, 7, 8, and 11)


Appendix 1 - Project Description

1 Description
This project involved researching, designing, implementing and evaluating strategic game AI techniques using StarCraft II Ai research environment. The student will design a new method for testing the environment. Because AlphaStar is not open for public yet, some goals/requirements of the project has been modified/removed. Specifically, students need to find a approach to analyze StarCraft II AI.

To achieve the goal, I read a great number of articles about existing AI research for StarCraft II, analyzed the statistics of human players. Then I analyze a AI from multiple aspects by some features. To prove these features are important to analyze AI. I build a classifier based on these features and it has a satisfying performance.

2 Learning Objectives

- Apply knowledge and implementation skills in computer science to develope and evaluate a strategy game AI.
- Deepen knowledge of advanced computing principles by examining, designing, and evaluating strategy game AI.
- Learn specific technical skills required for strategy game AI.
- Learn relevant project-related skills, including project management and oral and written communication, and apply these to project work.
INDEPENDENT STUDY CONTRACT
PROJECTS

Note: Enrolment is subject to approval by the course convenor

SECTION A (Students and Supervisors)

UniID: u6341895
Surname: Chen
First Names: Yinhe
g
Project Supervisor (may be external): Dr Penny Kyburz
Formal Supervisor (if different, must be an RISSC academic): Dr Penny Kyburz
Course Code, Title and Units: COMP4560, Advanced Computing Project, 12 units

Commencing Semester: S1 S2 Year: 2019 Two-semester project (12u courses only)

Project Title:
- Advancing strategic game AI in the StarCraft II AI research environment

Learning Objectives:
- Apply knowledge and implementation skills in computer science to develop and evaluate a strategy game AI
- Deepen knowledge of advanced computing principles by examining, designing, and evaluating strategy game AI
- Learn specific technical skills required for strategy game AI development and apply them to the project
- Learn relevant project-related skills, including project management and oral and written communication, and apply these to project work

Project Description:

This project will involve researching, designing, implementing, and evaluating strategic game AI techniques using the StarCraft II AI research environment. The student will research the StarCraft II AI research environment, previous work in the environment, and different AI approaches and design a new method for testing in the environment. Subsequently, the student will develop and evaluate the AI method and report on the results of their research.

This project will involve researching, investigating, and developing an understanding of: the
1. StarCraft II AI environment and previous work in this environment
2. Deep Mind’s AlphaStar StarCraft II AI
3. Deep Learning principles as applied to game AI
The student will be required to:
1. Set up and run tests using DeepMind’s AlphaStar AI in the StarCraft II AI Environment
2. Design game levels and scenarios to train and test the AlphaStar AI
3. Make minor modifications to AlphaStar AI where required, then train and test the AI


<table>
<thead>
<tr>
<th>Assessed project components:</th>
<th>% of mark</th>
<th>Due date</th>
<th>Evaluated by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report: style: Research report (e.g. research report, software description, ...)</td>
<td>60%</td>
<td></td>
<td>(examiner)</td>
</tr>
<tr>
<td>Artefact: kind: Software (e.g. software, user interface, robot, ...)</td>
<td>30%</td>
<td></td>
<td>(supervisor)</td>
</tr>
<tr>
<td>Presentation:</td>
<td>10%</td>
<td></td>
<td>(course convenor)</td>
</tr>
</tbody>
</table>

MEETING DATES (IF KNOWN):

STUDENT DECLARATION: I agree to fulfil the above defined contract:

\[\text{Signature}\] yin hong \[\text{Chen}\] \[26/02/2019\]

SECTION B (Supervisor):

I am willing to supervise and support this project. I have checked the student's academic record and believe this student can complete the project. I nominate the following examiner, and have obtained their consent to review the report (via signature below or attached email)

\[\text{Signature}\] \[27/02/2019\]

Examiner: \[\text{Jochen Renz}\]
Name: \[\text{Signature}\]
(Nominated examiners may be subject to change on request by the supervisor or course convenor)

REQUIRED DEPARTMENT RESOURCES:

\[\text{Signature}\]

SECTION C (Course convenor approval)

\[\text{Signature}\] \[29/2/2019\]
Appendix

Appendix 3 - Description of Artefact

In my project, I designed two main modules along with other small modules:

1. Model to decode StarCraft II replays

This model is constructed on top of s2protocol, it currently supports to decode 1 vs 1 StarCraft II replay, it contains two files:

- `analysis.py` reads StarCraft II replay file, you can load a replay like:

```python
from analysis import *
replay = analysis.SC2_replay('replay_path')
```

which returns a python class has following main methods:

1. `map_name()`: this method returns the name of the map of the match.
2. `player_details()`: this will returns a python dictionary, which contains players’ name, race, and match result. An example is shown below:

   ```python
   {'Player_1': {'Name': 'LiquidMaNa', 'Race': 'Protoss', 'Result': 'Loss'},
   'Player_2': {'Name': 'AlphaStar', 'Race': 'Protoss', 'Result': 'Win'}}
   ```

3. `resource_earning_details()`: this method returns a python nested dictionary, which contains details of players’ resource in each time index. An result fragment is shown below, where 228 is the time index in the game.

   ```python
   {'Player_1': {...,
   228: {'Current_Mineral': 35, 'Mineral_Collection_Rate': 895,
   'Current_Vespene': 258, 'Vespene_Collection_Rate': 335,
   'SQ': 82.5357737420317},...},Player_2':{...}}
   ```

4. `units_details()`: this method returns two python nested dictionary. The first dictionary contains the information about number and type of units, produced units, killed units for players in each time index. The second

---

1. [https://github.com/Blizzard/s2protocol](https://github.com/Blizzard/s2protocol)
Appendix 3 - Description of Artefact

dictionary contains the a series of events which tell players when their units born or die. The two dictionary have same data structure as that in last method.

5. `workers_num()`: this method returns a python nested dictionary, which shows the number of workers for players in each time index.

6. `Player_APM()`: this method return a python nested dictionary, which shows the players’ APM in each time index and their average APM through full game.

7. The returned results of other methods: `supply_details()` and `resource_used_details()` have same data structure as above methods.

- `example.ipynb` this file is an example of using `analysis.py` to decode a StarCraft II replay, analyzing and visualizing returned results.

2 Model to predict players/AI’s league

This model only contains one file `prediction.ipynb`, this file contains following parts:

1. `player_features()` function: this function uses `analysis.py` to extract and calculate all important features of a StarCraft II replay, these features include:

   - `League`
   - `Avg APM`
   - `Game length`
   - `Avg Mineral`
   - `Avg Mineral Rate`
   - `Avg Vespene`
   - `Avg Vespene Rate`
   - `Avg SQ`
   - `Avg Workers`
   - `Avg Supply Used/Made`
   - `Supply Blocked time`
   - `Produced Units`
   - `Killed Units`
   - `Killed Workers`
   - `Race`

   for players in a match.

2. A code fragment to extract above features from thousands of replays. This code fragment is commented out because it requires a large number of StarCraft II replays. You may not need to run this code fragment since I have done it and written these features to a dataset in `replay_dataset.csv` file, although the dataset maybe insufficient.

3. A part of pre-processing dataset (`replay_dataset.csv`), this includes:

   (a) Removing duplication and instances whose features are negative.
   (b) Replacing race name to integer.
   (c) Removing Bronze and Sliver instances since the they are extremely insufficient.
   (d) Removing instances in which game length is less than 2 minutes.
   (e) Normalisation.

4. A code fragment to train the classifier, and the result of classifier.

Once you finish above steps, you can use the classifier to predict AI’s league by its replays.
3 Others

Other files contains some necessary files to execute above modules, these includes:

1. nelson.py: a visualization module, this is used in example.ipynb.

2. a directory of replays for AlphaStar and profession players, these replays are used in example.ipynb

3. replay_dataset.csv for prediction.ipynb.

Other files for small work contain:

1. global_players.csv, PvP.csv (statistics of global players based on Protoss vs Protoss matches) and league_distribution.csv

2. global_players_data.ipynb: visualization of above files

3. a directory named scripts: this directory contains some python scripts to download statistics for global players and replays for Starcraft II.

4 Testing

The modules has no certain methods to test since fewer related work exist, however, you can always upload your own StarCraft II replays to visualise and predict result and compare result by watching replay in StarCraft II. Also, you can use sc2reader to compare features, because of different method to create timeseries, the value of features extracted by my module can be slightly difference with that extracted by sc2reader.

5 Acknowledgement

- s2protocol: version : 4.10.3.76114.0 https://github.com/Blizzard/s2protocol

- nelson.py: refers to https://github.com/IBM/starcraft2-replay-analysis/


- AlphaStar replays: https://deepmind.com/research/open-source/alphastar-resources

---

2Reference: https://github.com/IBM/starcraft2-replay-analysis/

3https://deepmind.com/research/open-source/alphastar-resources

4data from https://sc2replaystats.com/stats/analytics

5data from https://www.rankedftw.com/

6https://github.com/ggtracker/sc2reader/tree/upstream/sc2reader
• Statistics of global players: data retrieved from https://sc2replaystats.com/stats/analytics

• League distribution for global players: data retrieved from https://www.rankedftw.com/
Appendix

Appendix 4 - README

1 Description
Considering the fact that lack of method to analyze StarCart II AI, this project provides two modules which can be used to decode StarCart II replays, visualise players/AIs’ features in a match and predict AI’s league based on the features in StarCart II ranked game.

2 Requirement
These modules are encoded by Python3.7, they requires following packages, you can always use the newest versions of these packages. Other necessary packages are included in the artefact package.

- bokeh
- mpyq
- numpy
- pandas
- scikit-learn
- s2protocol

3 Models
The project have two main models, other files are necessary files to run following two models or they are designed for small work.

3.1 Model to decode StarCraft II replays
This model is constructed on top of s2protocol \(^1\), it currently supports to decode 1 vs 1 StarCraft II replay, it contains two files:

\(^1\)https://github.com/Blizzard/s2protocol
• **analysis.py** reads StarCraft II replay file, you can load a replay like:

```python
from analysis import *
replay = analysis.SC2_replay('replay_path')
```

which returns a python class has following main methods:

1. **map_name()**: this method returns the name of the map of the match.

2. **player_details()**: this will returns a python dictionary, which contains players' name, race, and match result. An example is shown below:

   ```python
   {'Player_1': {'Name': 'LiquidMaNa', 'Race': 'Protoss', 'Result': 'Loss'},
    'Player_2': {'Name': 'AlphaStar', 'Race': 'Protoss', 'Result': 'Win'}}
   ```

3. **resource_earning_details()**: this method returns a python nested dictionary, which contains details of players' resource in each time index. An result fragment is shown below, where 228 is the time index in the game.

   ```python
   {'Player_1': {...,
     228: {'Current_Mineral': 35, 'Mineral_Collection_Rate': 895,
           'Current_Vespene': 258, 'Vespene_Collection_Rate': 335,
           'SQ': 82.53557737420317},...},
   'Player_2': {...}}
   ```

4. **units_details()** this method returns two python nested dictionary. The first dictionary contains the information about number and type of units, produced units, killed units for players in each time index. The second dictionary contains the a series of events which tell players when their units born or die. The two dictionary have same data structure as that in last method.

5. **workers_num()**: this method returns a python nested dictionary, which shows the number of workers for players in each time index.

6. **Player_APM()**: this method return a python nested dictionary, which shows the players' APM in each time index and their average APM through full game.

7. The returned results of other methods: **supply_details()** and **resource_used_details()** have same data structure as above methods.

• **example.ipynb** this file is an example of using **analysis.py** to decode a StarCraft II replay, analyzing and visualizing returned results.

### 3.2 Model to predict players/AI's league

This model only contains one file **prediction.ipynb**, this file contains following parts:

1. **player_features()** function: this function uses **analysis.py** to extract and calculate all important features of a StarCraft II replay, these features include:

   ```
   'League', 'Avg APM', 'Game length', 'Avg Mineral', 'Avg Mineral Rate', 'Avg Vespene',
   'Avg Vespene Rate', 'Avg SQ', 'Avg Workers', 'Avg Supply Used/Made',
   'Supply Blocked time', 'Produced Units', 'Killed Units', 'Killed Workers', 'Race'
   ```
for players in a match.

2. A code fragment to extract above features from thousands of replays. This code fragment is commented out because it requires a large number of StarCraft II replays. You may not need to run this code fragment since I have done it and written these features to a dataset in `replay_dataset.csv` file, although the dataset maybe insufficient.

3. A part of pre-processing dataset (replay_dataset.csv), this includes:
   
   (a) Removing duplication and instances whose features are negative.
   (b) Replacing race name to integer.
   (c) Removing Bronze and Sliver instances since the they are extremely insufficient.
   (d) Removing instances in which game length is less than 2 minutes.
   (e) Normalisation.

4. A code fragment to train the classifier, and the result of classifier.

Once you finish above steps, you can use the classifier to predict AI’s league by its replays.

3.3 Others

Other files contains some necessary files to execute above modules, these includes:

1. `nelson.py`  
   : a visualization module, this is used in `example.ipynb`.

2. a directory of replays for AlphaStar and profession players, these replays are used in `example.ipynb`

3. `replay_dataset.csv` for `prediction.ipynb`.

Other files for small work contain:

1. `global_players.csv`, `PvP.csv` (statistics of global players based on Protoss vs Protoss matches) and `league_distribution.csv`  

2. `global_players_data.ipynb` : visualization of above files

3. a directory named scripts: this directory contains some python scripts to download statistics for global players and replays for Starcraft II.

---

2Reference: https://github.com/IBM/starcraft2-replay-analysis/
3https://deepmind.com/research/open-source/alphastar-resources
4data from https://sc2replaystats.com/stats/analytics
5data from https://www.rankedftw.com/
4 Acknowledgement

- s2protocol : version : 4.10.3.76114.0 https://github.com/Blizzard/s2protocol
- nelson.py : refers to https://github.com/IBM/starcraft2-replay-analysis/
- AlphaStar replays : https://deepmind.com/research/open-source/alphastar-resources
- Statistics of global players : data retrieved from https://sc2replaystats.com/stats/analytics
- League distribution for global players : data retrieved from https://www.rankedftw.com/