Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Statistical analysis of user-created game levels

Trabajo Final de Grado

Grado en Ingeniería Informática

Author: Álvaro Marco Añó

UPV supervisor: Roberto Paredes Palacios

ETH supervisor: Mubbasir Kapadia and Severin Klingler

September 3, 2014

Als que sempre han estat ahí.

Abstract

In some videogames, players have an enormous amount of maps to choose, the question that this thesis tries to answer is, what are the differences between a “good” map and a “bad” map? In this project a machine learning scheme was designed to try to answer this question for Minecraft maps.

Keywords: Machine learning, Data mining, Regression analysis, Game studies, Minecraft.

Resumen

En algunos videojuegos, los jugadores tienen a su alcance una enorme cantidad de mapas entre los que elegir, la pregunta que este TFG trata de responder es, ¿cuáles son las diferencias entre un “buen” mapa y un “mal” mapa? En este proyecto un esquema de aprendizaje automático se ha diseñado y aplicado para responder a esta pregunta en los mapas de Minecraft.

Palabras clave: Aprendizaje automático, Minería de datos, Análisis de la regresión, Ludología, Minecraft.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

This thesis was developed in collaboration of the CGL group[1] of ETH Zürich and Disney Research Zürich, its final purpose is to find out if it possible to predict the popularity of user-created videogame levels and obtain conclusions about user preferences on those levels. For the research project, Minecraft has been chosen as a game to analyze; but the idea is to generalize this work in the future.

The programming language used during the development of the project is Java and some external libraries were used as well. Octave was used for plotting purposes.

In this first chapter the aim and motivation of the thesis are presented, then in the second section context and features about Minecraft are described.

[1] Computer Graphics Laboratory

1.1. Aim and motivation

Nowadays videogames have a huge impact in society culturally and economically, this market generates billions of dollars of revenues worldwide and there are even professional videogame leagues in some countries. Because of this, analyzing videogames components such as maps, elements that form the maps, different artificial intelligence approaches for enemies, game modes, players behaviour and players interaction between them becomes a very interesting field [Bai07].

Game studies have applications in many areas such as computer science, psychology, anthropology, communications, etc. As a concrete example imagine the case in that videogame contents are analyzed so that they can be improved; this can lead to higher profits to videogame companies among many other applications.

More concretely, in this thesis a machine learning pipeline has been developed to try to predict the popularity of Minecraft levels. Analyzing which maps are popular and which maps are not gives us the possibility to detect the features that define a popular map. This information is very useful since it can be used to develop assistants for level cration, or even more, generating levels of high quality from scratch.

As previously said, Minecraft has been chosen for this thesis, but why? The reasons are presented in the next section.

1.2. Minecraft

Minecraft[2] is an open world videogame, in which players have considerable freedom in terms of what actions to perform and and what to do in general. Due to the freedom that the game provides it is a perfect piece of analysis since it does not limit the player in terms of where to go or what to do while playing. Even more, due to this characteristic, the results of the analysis could even be used for more freedom-limited videogames as well as other open world games (e.g. Disney Infinity).

However, even though the game was release in 2009 it is still under development and new features and improvements are added regularly (in fact

[2] released on 2009 and developed by Mojang AB.

the last update, version 1.8, was released on September 2, 2014).

But how does Minecraft work? In the game everything is shaped as a block (or cube) and players can explore the map, destroy blocks (mine) for gathering resources, craft new items (tools, materials, furniture, weapons, etc. more than 100 items can be crafted), combat, build constructions and much more. As it can be seen in [Yee], a good crafting system is found important for many players. Like so, players interested in building, fighting, exploring or a mix of everything can also find a motivation in playing the game.

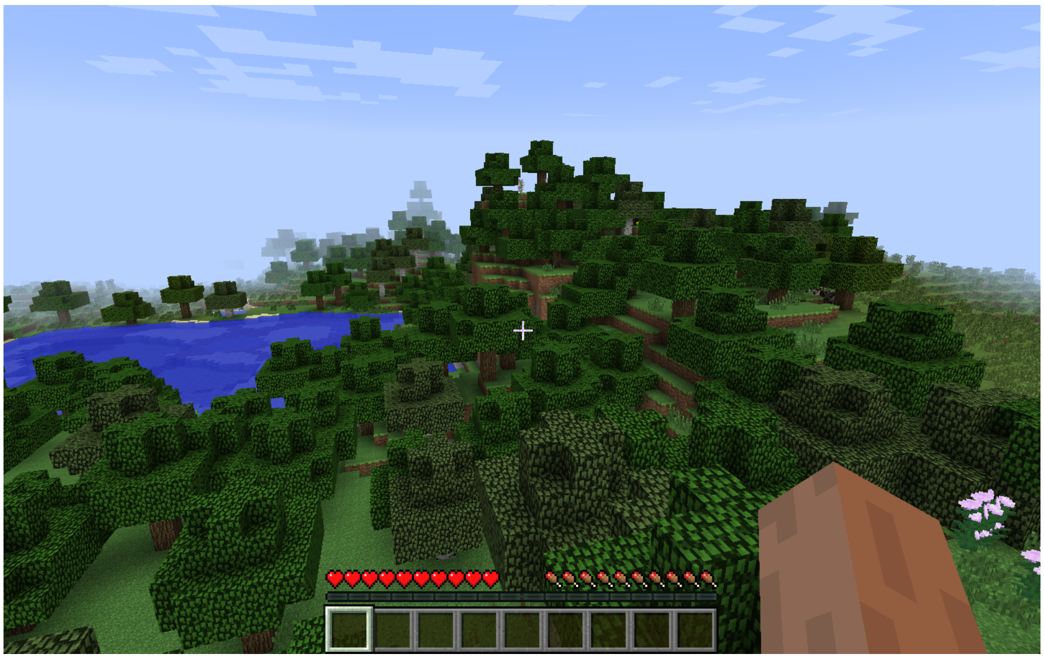
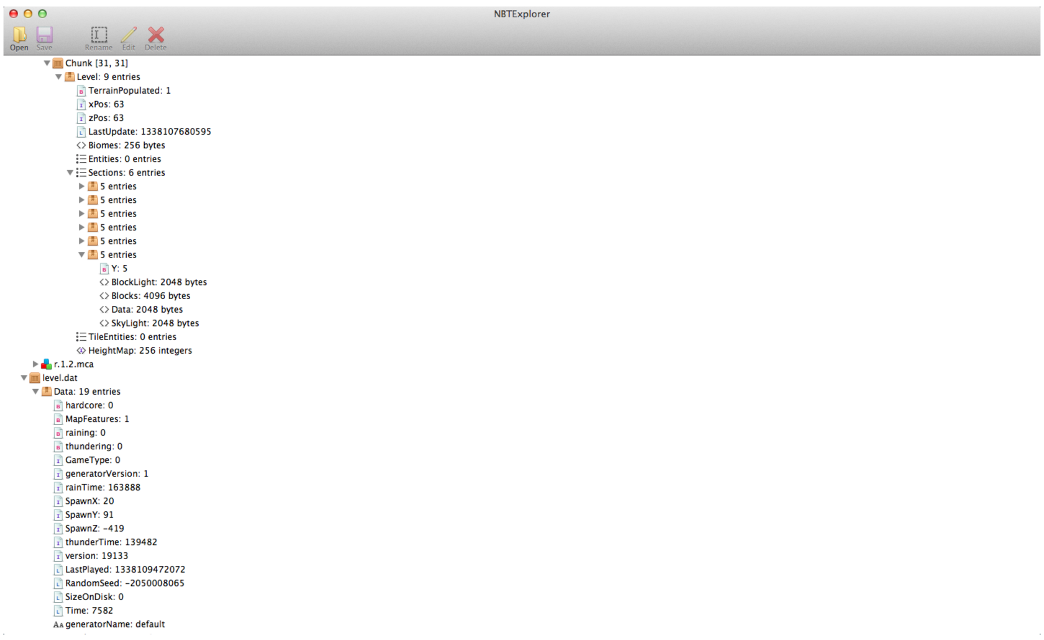


Figure 1.1: Automatically generated world in survival mode.

Mojang's game is the best selling PC game of all time, it has sold more than 16 million copies only on its PC/Mac version, for this reason there is a huge community of players and fans playing and creating content which leads to a perfect situation for analyzing the game. On the internet there are thousands of websites dedicated to the Minecraft and hundreds of them sharing user-created content.

Since the game was released, a lot of people was interested on how the game worked and at this point there are a lot of resources available in this sense. The community did apply techniques such as reverse engineering to know the functioning of the game, but also Mojang developers post frequently in a blog and give crutial information about the game mechanics. There are tools available for rendering and visualizing maps (both in 2D and 3D), libraries for reading and writing Minecraft formats, map editors, map generators, texture editors, backup utilities and so on.



```
▼ Chunk [31, 31]
  ▼ Level: 9 entries
      TerrainPopulated: 1
      xPos: 63
      zPos: 63
      LastUpdate: 1338107680595
      Biomes: 256 bytes
      Entities: 0 entries
    ▼ Sections: 6 entries
      ▶ 5 entries
      ▶ 5 entries
      ▶ 5 entries
      ▶ 5 entries
      ▶ 5 entries
      ▼ 5 entries
          Y: 5
          BlockLight: 2048 bytes
          Blocks: 4096 bytes
          Data: 2048 bytes
          SkyLight: 2048 bytes
      TileEntities: 0 entries
      HeightMap: 256 integers
▶ r.1.2.mca
▼ level.dat
  ▼ Data: 19 entries
      hardcore: 0
      MapFeatures: 1
      raining: 0
      thundering: 0
      GameType: 0
      generatorVersion: 1
      rainTime: 163888
      SpawnX: 20
      SpawnY: 91
      SpawnZ: -419
      thunderTime: 139482
      version: 19133
      LastPlayed: 1338109472072
      RandomSeed: -2050008065
      SizeOnDisk: 0
      Time: 7582
      generatorName: default
```

Figure 1.2: NBTExplorer, a tool for reading and writing NBT files (to be introduced later) showing the general structure of a world.

On the "player-side" there exist websites with even more than 30000 maps freely available for downloading, websites dedicated to mods that change the mechanics of the game, free servers for online gaming and an ongoing list of material. Moreover Mojang has always encouraged the creation and sharing of this type of materials and they even planned to release a Plugin API to allow developers to easily add new content to the game, nonetheless this API is still under development, but when available it will allow the creation of plugins without modifying the `minecraft.jar` executable.

Minecraft has five different game modes (without the use of any mod): adventure, survival, hardcore, creative and spectator. Each one of this modes has different objectives, for instance in the creative mode access to an infinite amount of almost all blocks is provided, blocks can be destroyed instantly and players are invulnerable (they cannot die). In survival mode players have to acquire resources to build, craft and gain experience and also they have to maintain their health and hunger bars. Adventure mode is very similar to survival mode but in this mode blocks can only be broken with the correct tools and there are even blocks that cannot be destroyed to avoid spoiling the adventure map. This thesis is focused on the analysis of advenutre maps.

In the game, worlds can be automatically generated by using a seed and a level generator. Another widespread option is to generate an automatic map and then build structures on it, locally or collaboratively online, and finally share the map through one of the many websites on the Internet dedicated to this aim. This thesis analyzes user-created game levels available on the Internet.

For all of these reasons Minecraft was chosen to be analyzed in this thesis.

CHAPTER 2

POPULARITY PREDICTION FOR MINECRAFT

In this chapter the machine learning pipeline used for the popularity prediction of Minecraft levels is detailed.

In the first section, an overview of the pipeline is given and in each of the following sections one of the components of the pipeline is introduced. The second section describes the process for acquiring the Minecraft maps and shows statistics on the collected data; in the third section the designed features are detailed and finally in the last section the learning process is presented.

2.1. The pipeline

The pipeline followed during the development of the analysis covers the entire process of a machine learning schema and is formed by three main components, which are subdivided as well into smaller subcomponents. This problem does not have such a big research investment so far as other problems, such as speech recognition or OCR[1]. In these other mentioned problems

[1]Optical Character Recognition

with many years of research there exist a clear set of features that are well-known and work well for their analysis and classification, however this is not the case of the problem presented in this thesis. For this reason, the main effort of the research project was put on the design and modelling of the features.

The first implemented step in the pipeline consists in the acquisition (and preprocessing) of a set of Minecraft maps for their further analysis. Next, maps are parsed and features extracted from them. The last step consists on applying learning techniques to predict the popularity of the maps. After that, iterations over this last two steps are needed to check how well the predictions are functioning and devise new features to improve the results.

The diagram in figure 2.1 pictures the described pipeline.

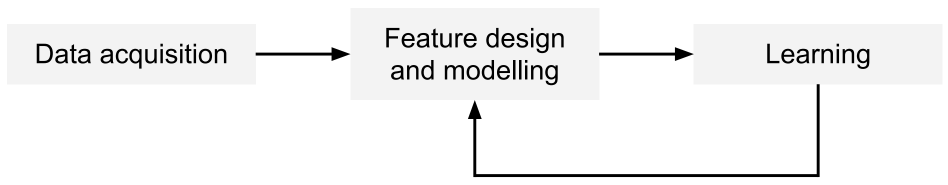


Figure 2.1: Diagram displaying the pipeline used during in the analysis of features.

Moreover, some of this stages are subdivided into smaller substages, but this substages will be presented in the corresponding section.

Once the features provide consistent results the pipeline in figure 2.2 may be used. This is the standard machine learning approach, in which a model is built using a training set and afterwards this model is used for performing predictions on the testing or “real-world” data.

In the next sections the elements of the pipeline are described in detail.

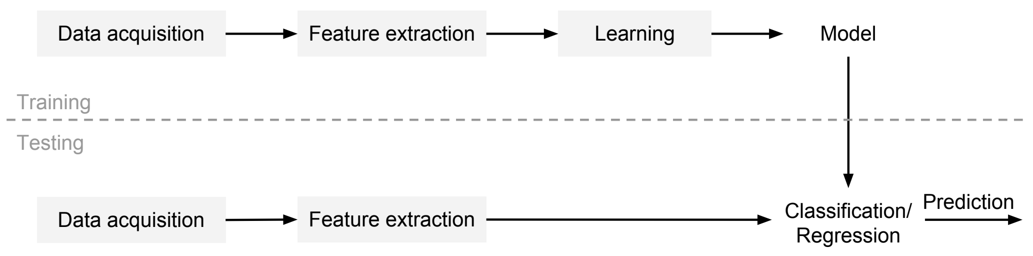


Figure 2.2: Diagram displaying the standars approach followed adopted for dealing with machine learning problems.

2.2. Acquiring the data

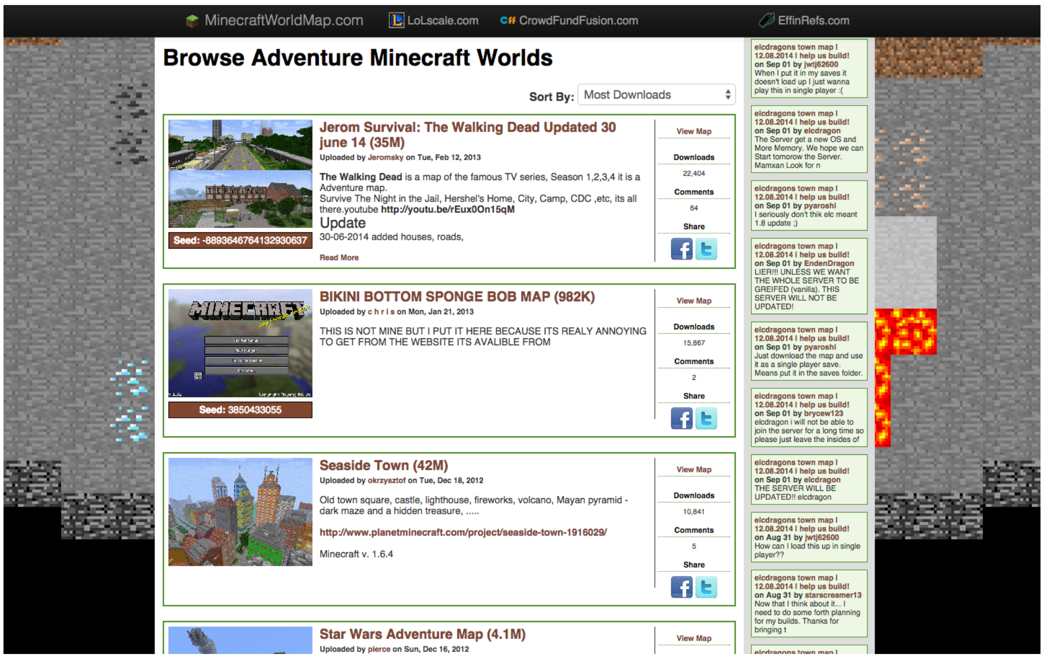
Even though Minecraft has a huge community of people behind it Mojang does not offer an official website for sharing the worlds, mods, textures or plugins created by the users. Recently, Minecraft Realms was released but this service only offers players to create and manage their own private server and there is not a world-sharing service so far. However there exist lots of unofficial websites, some of them with tons of material and some of them (most) with not that much.

There is not a clear structure among these unofficial websites in terms of the meta-information that they provide. The meta-information provided in the websites varies from one website to another but this is a short list of some of the offered data:

- Number of downloads and comments
- Raw comments
- Author of the map
- Upload and update date
- Map version
- Description of the map
- Topics
- Map size

- Ratings, likes, favorites and other social-network-related information
- Screenshots
- ...

When choosing a website for downloading the levels two considerations were taken into account. First the website needed a quite big amount of levels, and second the maps had to be directly available for download from that website. Many websites do not allow a direct download, but they offer a link to an upload server from where you can download the level. This situation is problematic because it complicates the automatic download of the levels (each website has a different procedure for downloading: waiting one minute, captchas, etc). A website that fulfills both requirements is the website `http://www.minecraftworldmap.com/` which was the one finally chosen for downloading the levels.



Browse Adventure Minecraft Worlds

Sort By: Most Downloads

Jerom Survival: The Walking Dead Updated 30 june 14 (35M)
Uploaded by Jeromsky on Tue, Feb 12, 2013
The Walking Dead is a map of the famous TV series, Season 1,2,3,4 it is a Adventure map.
Survive The Night in the Jail, Hershel's Home, City, Camp, CDC ,etc, its all there.youtube **http://youtu.be/rEux0On15qM**
Update
30-06-2014 added houses, roads,
Read More
Seed: -8893646764132930637
Downloads 22,404
Comments 64

BIKINI BOTTOM SPONGE BOB MAP (982K)
Uploaded by c h r i s on Mon, Jan 21, 2013
THIS IS NOT MINE BUT I PUT IT HERE BECAUSE ITS REALY ANNOYING TO GET FROM THE WEBSITE ITS AVALIBLE FROM
Seed: 3850433055
Downloads 15,867
Comments 2

Seaside Town (42M)
Uploaded by okrzysztof on Tue, Dec 18, 2012
Old town square, castle, lighthouse, fireworks, volcano, Mayan pyramid - dark maze and a hidden treasure, ...
http://www.planetminecraft.com/project/seaside-town-1916029/
Minecraft v. 1.6.4
Downloads 10,841
Comments 5

Star Wars Adventure Map (4.1M)
Uploaded by pierce on Sun, Dec 16, 2012

elcdragons town map I 12.08.2014 I help us build! on Sep 01 by jwtj62600
When I put it in my saves it doesn't load up I just wanna play this in single player :(

elcdragons town map I 12.08.2014 I help us build! on Sep 01 by elcdragon
The Server get a new OS and More Memory. We hope we can Start tomorow the Server. Mamxan Look for n

elcdragons town map I 12.08.2014 I help us build! on Sep 01 by pyaroshi
I seriously don't thik elc meant 1.8 update ;)

elcdragons town map I 12.08.2014 I help us build! on Sep 01 by EndenDragon
LIER!!! UNLESS WE WANT THE WHOLE SERVER TO BE GREIFED (vanilla). THIS SERVER WILL NOT BE UPDATED!

elcdragons town map I 12.08.2014 I help us build! on Sep 01 by pyaroshi
Just download the map and use it as a single player save. Means put it in the saves folder.

elcdragons town map I 12.08.2014 I help us build! on Sep 01 by brycew123
elcdragon i will not be able to join the server for a long time so please just leave the insides of

elcdragons town map I 12.08.2014 I help us build! on Sep 01 by elcdragon
THE SERVER WILL BE UPDATED!! elcdragon

elcdragons town map I 12.08.2014 I help us build! on Aug 31 by jwtj62600
How can I load this up in single player??

elcdragons town map I 12.08.2014 I help us build! on Aug 31 by starscreamer13
Now that I think about it... I need to do some forth planning for my builds. Thanks for bringing t

Figure 2.3: Screenshot of the website `http://www.minecraftworldmap.com/`.

The selected website has more than 30000 maps available for download and also is quite meta-information rich. However it does not provide any

API or system to allow an easy download of the maps and meta-information, consequently a web crawler was developed in order to download the maps and its corresponding information. The crawler works as follows: it downloads a webpage, opens each world link in that webpage and stores the world and its meta-data. In this way, a big number of levels can be downloaded in an easy way just by iterating over the sections in the website. The library jsoup was used as a help for parsing the html files.

Each uploaded level has the following associated information available directly on the website: number of downloads, number of comments, raw comments, author, upload date, size, description, topics and a render of the map navigable through Google Maps. However not every piece of this information can be used easily, for example comments need to be parsed and analyzed separately and the results of doing this work are uncertain. Due to that the information stored is: number of downloads, number of comments, author, upload date and size.

For storing the meta-information a database was designed and implemented using SQLite. In this database not only the meta-information is stored, but also the features extracted from the worlds.

In the next subsection some descriptive statistics on the downloaded levels are presented.

2.2.1. The data

When downloading the maps some preprocessing was made to check that the downloaded file was alright and actually contained the map (uncompressing the file, checking the existance of the files, etc). After this preprocessing a total of 1717 were downloaded (all the levels that do not contain errors in the category “adventure” of the website). This is a total of 19.5 GB, with an average file size of 10 MB and with the “biggest” level with a weight of 420 MB.

Table 2.1 shows how the number of downloads is distributed among the levels: the first column is the number of downloads, the second column the number of levels that have at least that number of downloads and the third column the percentage of the total number of levels that were downloaded at least that given number of times.

Table 2.1: Distribution of the number of downloads.

Number of downloads	*Number of levels*	*Percentage*
1	136	7.9 %
10	773	45.0 %
50	1278	74.4 %
80	1379	80.3 %
100	1419	83.0 %
200	1537	89.5 %
500	1623	94.5 %
1000	1660	96.7 %
10000	1711	99.7 %
100000	1717	100.0 %

Not surprisingly, there are a lot of maps that have very few downloads (three quarters of the worlds have 50 or less downloads!) and a few fraction of the maps receives most of the attention. This can also be seen in figure 2.4.

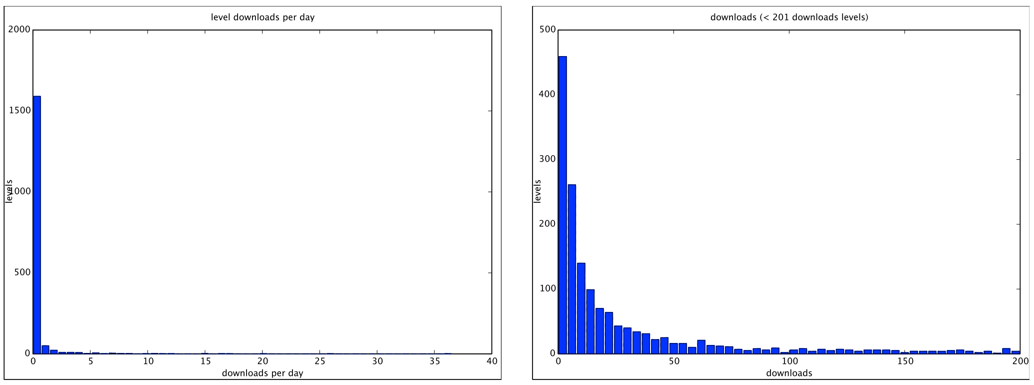


Figure 2.4: Barchart of normalized number of downloads (left) and barchart of number of maps with less than 201 downloads (right).

A similar situation occurs with the number of comments; most maps have only a few comments, but now the case is more extreme since 1696 maps (98.7%) have 5 or less comments. For this reason the focus is put in the number of downloads because the comments on the used website are not extensively used.

On the other hand, the topic tagging system of the website does not impose any restictions on the number or format of these topics. Due to this reason, things like worlds without any topic, worlds with 20 topics or topics like "minecraft" or "asdfasdfasdf" are common. This leads to a similar situation as in the raw comments, in order to take some advantage of the topic system some preprocessing has to be done first for filtering and mapping the topics. This context is almost identical to the raw comments one, therefore the topics were not considered when downloading the meta-data.

2.3. Features

To extract the features out of the Minecraft levels, they must be read and parsed; Minecraft has its own file format for the various files in which it stores the data. This file format is the NBT[2] format and it is formed by a tree structure made up of various tags [Per]. Moreover this information is arranged in a way that makes easier the compression of the files (which is made in the gzip format). The library jnbt was used for parsing the NBT files and some other tools were developed as well to facilitate this task.

In this section the designed features are listed and described detailedly. First the low-level features, which are more simple are introduced; after that the high-level features are presented.

2.3.1. Low-level features

The low-level features are features contained in the world file structure itself or features that are easy to obtain after parsing a map (e.g. counts, or averages).

Here it is an extensive list of the low-level features extracted from the meta-information files:

Generator of the level

The features obtained from the generator of the level:

Generator type

Possible values are: default, flat, largeBiomes, amplified or customized. Each one of them produces a different kind of level.

[2] Named Binary Tag

Generator version

Generator options
A set of extra options can be specified for the generator of the level.

Map features
If structures such as villages, strongholds and mineshafts should be placed or not.

Game rules
This includes some predefined behavior in the game. Rules for controlling if mobs (enemies) spwan naturally and wether the player keeps the inventory when dying are some of the possible rules among many others.

Seed
The seed used for the level generation.

The game mode
Survival, creative, spectator…

Difficulty
Four different difficulty levels are available and also whether the difficulty is locked.

Village-related features
Villages and other map features (temples, strongholds, mineshafts…) have its own information. As a way of illustration, villages have the next features that were stored for each village:

Number of golems in the village
Number of iron golems that protect the village from hostile enemies.

Number of villagers
Number of villagers (NPC[3] characters that can trade items with the player).

Location of the village
x, y, z coordinates of the center of the village.

Radius of the village
i.e. how big the village is.

[3] Non-player character

Number of doors in the village
Which is an approximation of the number of buildings.

A part from that meta-information other low-level features are extracted from the block information directly. These features include:

Count of each type of block
In practice it is normalized divided over the number of total blocks in the world.

Entropy of each type of block

Count of each type of biome
Biomes are areas in Minecraft worlds with varying geographical features: height, temperature, flora, etc. Examples of biomes are: plains, forests, beaches, rivers, jungles, oceans...

Total number of blocks

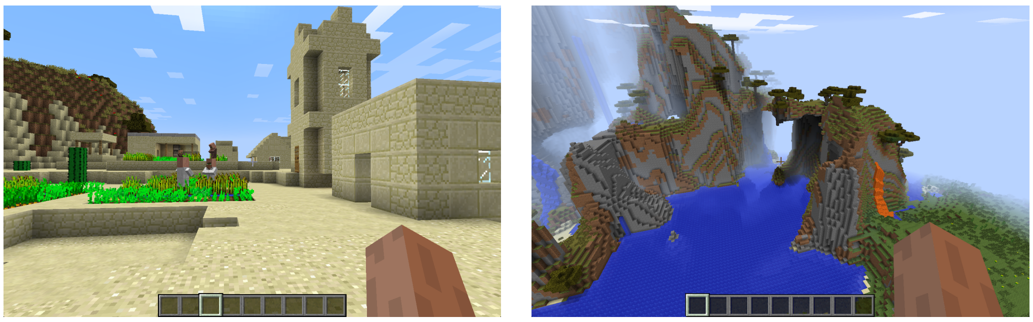


Figure 2.5: Screenshots of a village with NPCs (on the left) and different types of biomes (on the right).

This information is extracted for each one of the three dimensions that may exist in a Minecraft level: The Overworld ("normal" game world), The Nether (hell-like dimension) and The End (a dimension formed by a single floating island). Also notice that every block is identified by 8 bits and so there are 256 (2^8) types of blocks. This leads to a big amount of features, in the same way there are also 256 possible types of biomes.

After the design and implementation of the mentioned low-level features new more complex high-level features were proposed and implemented with the aim to improve the predictor performance. These features are described in the next subsection.

2.3.2. High-level features

In the current section, high-level features are described. In contrast to low-level features, high level features require more work to get and think about.

The first designed high-level features consist on extracting high-level information from the blocks. For example finding geographical elements in a map, such as mountains, forests, oceans or deserts, and its corresponding features: height of a mountain and materials that form it, number of trees in a forest, ocean deepness, etc. For some of this features the problem can be simplified from 3 dimensions to 2 dimensions, this is the case of mountains and forests for example.

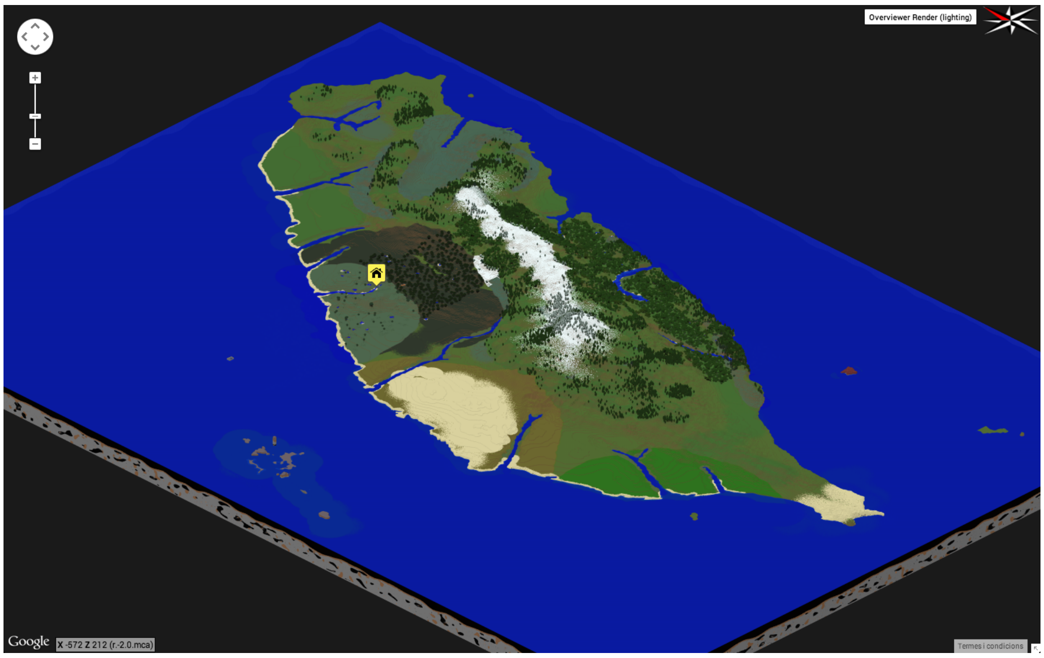


Figure 2.6: A map rendered in Google Maps.

The approach used for doing this 3D to 2D transformation consists on parsing each (x, z) coordinate of a map from top to bottom until finding the desired block type. Logs can be the goal when looking for a forest, the first "ground" block when looking for mountains, etc. An example is presented next.

The map in figure 2.6 contains a big island with forests, beaches, mountains, rivers and a few small islands surrounding the big one. After applying the described procedure to a map, heat maps can be obtained as the ones in figure 2.7.

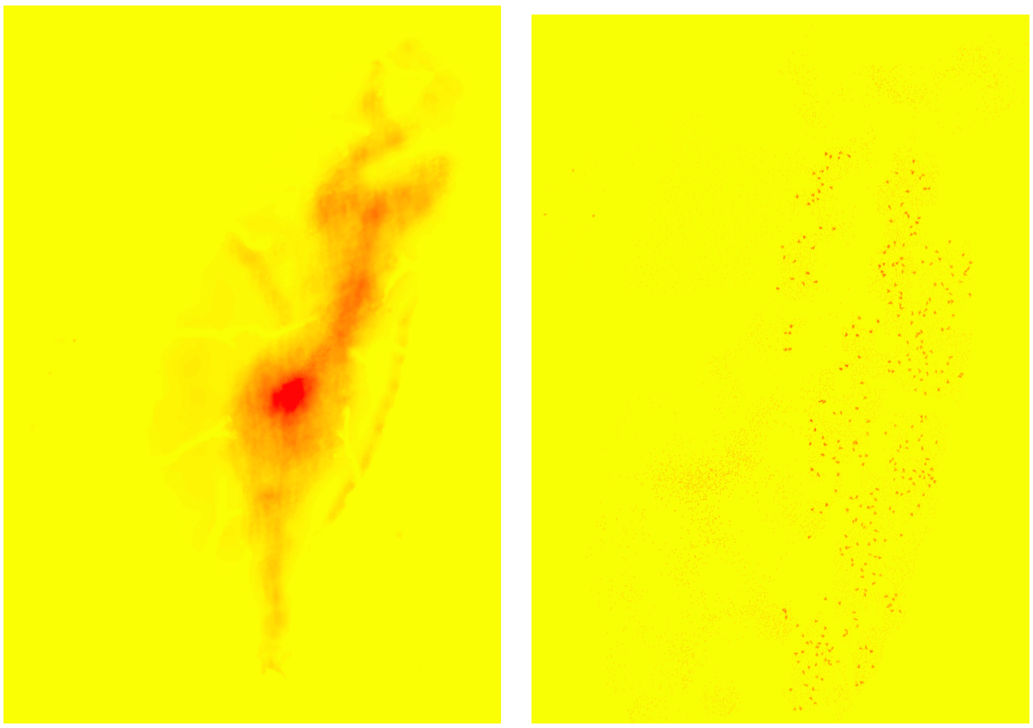


Figure 2.7: Heat maps of height (left) and logs (zoomed, right) of the world in figure 2.6.

Once we have a 2-dimensional representation of a map, other algorithms can be applied for further feature extraction. In the case of finding forests the K-means algorithm can be executed, nonetheless this is not as easy as that because the number of clusters K has to be specified to the algorithm, and it is unknown beforehand.

Finally, some of the extracted high-level features are listed below:

Variance of the height of terrain

This can be a hint of how "accessible" is the map.

Number of forests in a world

Average number of trees in a forest

Variance of the number of trees on each forest

After describing the features designed and implemented, it is time to see how the learning was modelled.

2.4. Learning

Once we have the data and a set of features to begin with it is time for building a model that predicts the popularity of a level. The popularity of a level is defined as the normalized number of downloads that the level has. To normalize the popularity this formula is used:

$$normalized\_popularity = \frac{number\_downloads}{upload\_day - today} \tag{2.1}$$

Note: the number of downloads must be updated to today's date!

Another possible approach is to introduce intervals for "good", "medium" and "bad" quality worlds, tag each sample accordingly depending on its number of downloads and transform the regression problem into a classification problem.

Now that the prediction objective has been introduced the learning scheme will be presented. For this task the machine learning library weka for Java has been used, a library with more than twelve years of development [HFH+09].

The whole weka learning scheme is sketched in figure 2.8, and is described in the next paragraphs step by step.

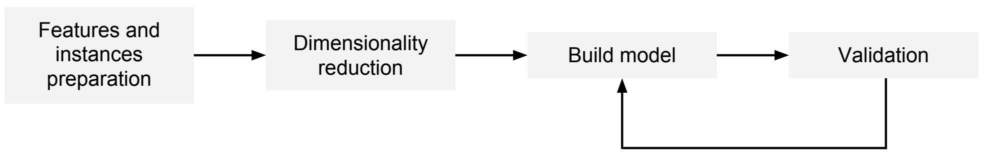


Figure 2.8: Scheme of the weka learning process used.

First of all a set of feature must be declared, for this purpose an array of `Attribute`s has to be created and each element of this array will be a level feature with one of the weka types: numeric, nomintal, date, string or relational. In this problem only numeric and nominal types are used. When the `Attribute`s are set up a object of the class `Instances` is declared. Then, an iterative process is carried out and on each step a new `Instance` is added to the `Instances` object; in our case each `Instance` or sample is composed by the features described in the previous chapter and it represents a Minecraft world. At the poing in which each `Instance` is created convenient preprocessing and normalization are introduced, when apply.

After the instances are prepared, dimensionality reduction is performed. The dimensionality reduction is done using the PCA[4] algorithm implemented in weka in the `PrincipalComponents` class. This step is essential, since the number of features is very large originally and it must be reduced in order to avoid the "curse of dimensionality" that arises when analyzing data in high-dimensional spaces.

Once the samples are prepared and the dimensinality has been reduced a model can be built. Weka disposes of several classifiers, regressors and many other tools [FHH+05]. Different learning models were tried for learning a regression predictor from the dataset. Classes like `LinearRegression` and `SMOreg` from the weka library were tested. The class `SMOreg` which implements SVMs[5] for regression models is of special interest [SS98]. Different parameters and kernels were testes were tested too.

Next, in order to validate the built model k-fold cross-validation is performed on the data. This method consist in splitting the data in k-folds, train the model in $k - 1$ folds and test the trained model in the remaining 1-fold set of samples; consequently k models are trained. In our scenario k is set to 10 [Koh95].

The results obtained out of this process are presented in the next chapter.

[4] Principal Component Analysis
[5] Support Vector Machine

CHAPTER 3 RESULTS

The results obtained after applying the process described in the second chapter of this document are analyzed in the following section.

3.1. Analysis of the results

As previously mentioned, different regressors were tested for solving the problem. The regressor that obtained the best results is the SVM with a huge difference. Using this regressor, different parameters need to be set, such as the kernel function or the regularization paramters.

Two different kernel functions were tried: the polynomial kernel (`PolyKernel`) and the RBF[1] kernel (`RBFKernel`).

The polynomial kernel is defined as:

$$K(x,y) =< x, y >^p \tag{3.1}$$

And it has as a parameter the exponent and the regularization parameter C.

[1] Radial Basis Function

On the other hand, the RBF kernel function is defined in this way:

$$K(x,y) = e^{-(\gamma < x-y, x-y >^2)} \tag{3.2}$$

The two parameters of this kernel function are γ and the regularization parameter C.

First, the polynomial kernel was tested with different values for the exponent (and some variations for C); clearly the best exponent was 1. Table 3.1 contains the results obtained.

Table 3.1: SVM regression with polynomial kernel.

Total error	*Average error*	*Variance*	*C*	*Exponent*
5726.065	58.900	2046447.385	1	1
2665.579	27.441	527407.587	0.1	1
163.163	1.669	140.656	0.01	1
330.996	3.403	5131.735	0.001	1
72482417.220	747222.895	5.39E+9	0.1	2
1239.186	12.760	117844.233	0.01	2

It can be seen in the table 3.1 that the best model trained with the polynomial kernel got an average prediction error on the number of downloads per day of 1.669 and it has a variance of 140.656.

After observing and analyzing the results of the polynomial kernel, the RBF kernel was tested. As in the case of the polynomial kernel, the best results come when finding the appropiate values for the parameters. Table 3.2 shows the results obtained with this kernel function.

Table 3.2: SVM regression with RBF kernel.

Total error	*Average error*	*Variance*	*C*	γ
68.62669971	0.702951778	5.01650386	1	0.1
60.04567207	0.615084691	5.02906642	0.1	0.1
57.30524711	0.587013491	5.14551194	0.001	0.1
57.01203592	0.584006570	5.20180667	0.0001	0.1
62.20121474	0.637095065	5.08452111	1	0.01
59.15346329	0.605910599	5.13774146	0.1	0.01
57.17557337	0.585678120	5.19795612	0.001	0.01
62.20121474	0.637095065	5.08452111	0.0001	0.01
60.08830397	0.615470567	5.17649649	1	0.001
57.16230031	0.585541728	5.20004569	0.1	0.001
56.94613503	0.583329872	5.22294997	0.001	0.001
56.99820582	0.583864299	5.23160565	0.0001	0.001

It can be observed in the table that the best model was obtained using the parameters C=0.01 and $\gamma = 0.001$. Figure 3.1 plots the values on table 3.2.

The results obtained for the best model on each one of the folds of the cross-validation is presented in the table 3.3.

Table 3.3: 10-fold results for the RBF kernel with C=0.01 and $\gamma = 0.001$.

fold	*Total error*	*Average error*	*Variance*
1	20.59717227	0.210175227	0.193403736
2	133.4066447	1.361292293	17.95626855
3	31.13547913	0.317708970	0.508038573
4	55.72015753	0.568573036	3.963004249
5	46.62647497	0.475780356	1.248203319
6	36.74163923	0.374914686	1.317500484
7	31.64706606	0.322929245	0.890236494
8	69.29391184	0.714370225	5.768442868
9	92.87732111	0.957498155	18.17760173
10	51.41548349	0.530056530	2.206799699

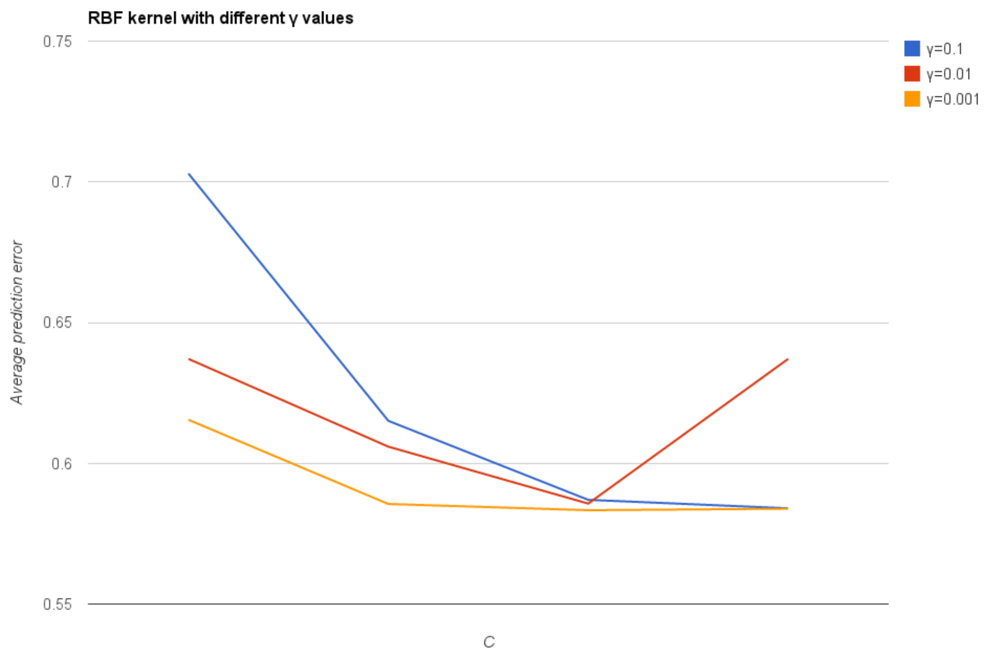


Figure 3.1: Average prediction error of the RBF kernel using different values of γ and C.

To conclude with this chapter and after studying the data presented, it can be stated that for this problem the best regression predictor is clearly a SVM regressor with an RBF kernel.

CHAPTER 4
CONCLUSIONS

In this work the problem of predicting the goodness of Minecraft levels was introduced and a machine learning scheme was designed and implemented to accomplish the goal. Finally the results obtained were analyzed carefully. The task of predicting the popularity of user-created game levels is quite complex, however the results obtained were quite good.

In the following section future work to be done is presented briefly.

4.1. Future work

Future research work for continuing the project could include the design of new high-level features. Some examples are given in the next paragraphs.

A good starting point could be implementing other high-level features that take into account not only 2D structures but also 3D ones. For example extracting features of caves in the worlds or even automatize and generalize the process of feature extraction could be done. However a smart preprocessing should probably be done before applying any learning technique because the number of blocks in some maps is enormous, in the order of hundreds of millions, but even higher in some worlds.

As a way of illustration, another completely different set of high-level features could be obtained by performing diffs between the downloaded levels and the corresponding "original" ones, that may be generated using the seed of the level. By doing this operation the position and kind of the blocks that were placed, changed or removed can be known. In this way, the "work" done by the players can be used as a new set of features.

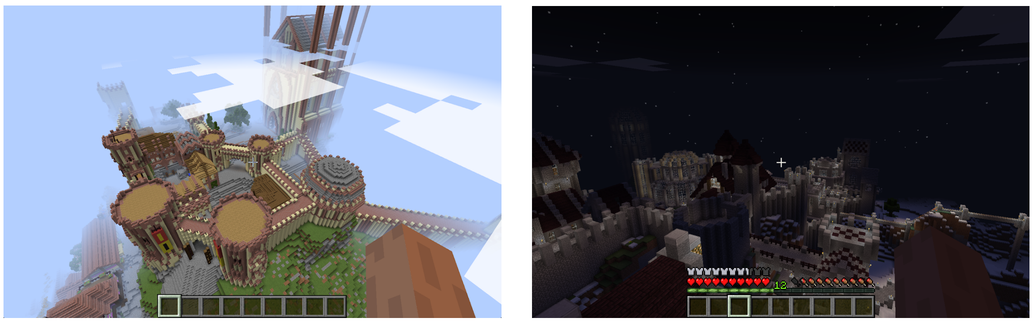


Figure 4.1: Screenshots of user-created levels with complex constructions.

Finally, parsing the raw comments and the topics of a world using NLP[1] tools would provide a good set of features to be tested as well. After doing this, conclusions as knowing if the players take into account the topics for downloading a world or if "zombie" maps are more popular could be obtained.

On the other hand, an obvious next step to perform is to generalize the scheme, try it with other videogames and observe the results.

[1] Natural Language Processing

BIBLIOGRAPHY

[Bai07] William S. Bainbridge. The scientific research potential of virtual worlds. *Science*, 317(5837):472–476, July 2007.

[FHH$^+$05] E. Frank, M. A. Hall, G. Holmes, R. Kirkby, B. Pfahringer, and I. H. Witten. *Weka: A machine learning workbench for data mining.*, pages 1305–1314. Springer, Berlin, 2005.

[HFH$^+$09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[Koh95] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[Per] Markus Alexej "Notch" Persson. Named binary tag specification. http://web.archive.org/web/20110723210920/http://www.minecraft.net/docs/NBT.txt. [Online; accessed 31-August-2014].

[SS98] A.J. Smola and B. Schoelkopf. A tutorial on support vector regression. Technical report, 1998. NeuroCOLT2 Technical Report NC2-TR-1998-030.

[Yee] Nick Yee. Crafting and trading from "The Daedalus Project". http://www.nickyee.com/daedalus/archives/001629.php. [Online; accessed 2-September-2014].