BGG-Analysis Documentation

1 Requirements

In order to build the database you will need Python version 3.6.1 or later, 64-bit or higher. In order to search the database, you will need Python version 3.6.1 or later. Use as a library may or may not require a 64-bit version of Python, depending only on the size of the database in use.

Python will install the following packages (using `pip`) as well as all of their dependencies in order to build or search the database:

Package	Build	Search
`requests`	✓	
`pandas`	✓	✓
`scikit-learn`	✓	✓
`unidecode`		✓
`jsonschema`		✓
`multiset`		✓

Please be aware that the `scikit-learn` package requires `scipy`, which does not have a functioning wheel for the default Python installation on Windows. As such it is recommended that you use WinPython instead, which includes `scipy` be default.

Running this code will also make appropriate use of packages `ujson` and `tqdm` if they are installed, but they are not required.

2 Location

The project `bgg-analysis` in its current state can be found at `https://github.com/Jonah-Horowitz/bgg-analysis`.

3 Installation

To install, place the following files in the same folder:

```
bgg_prepare.py
bgg_search.py
query_schema.json
```

If you already have a copy of the database, place the following two files in the above folder as well:

```
boardgames-clean.sqlite
ratings_svd.pickle
```

If you do not yet have a copy of the database or plan to build one yourself, place the following files in the above folder as well:

```
bgg_clean.py
bgg_collect.py
bgg_optimize.py
```

4 Building the Database

To build the database, run `bgg_prepare.py` - make sure your machine has an active internet connection. This will likely take quite a while. If it is disrupted, there is likely a way to resume approximately where it left off, but the method of doing this is dependent upon which stage of data collection/cleaning it was in when it was disrupted.

5 Searching the Database

Searching the database can be done in two related ways.

- If you are running `bgg-analysis` as a stand-alone program, you will need to create a file named `input.json` in the appropriate format (see later in this section), then run `bgg_search.py`.
- If you are using `bgg-analysis` as a Python library, you will need to call the function `bgg_search.process_query` with one argument. This argument can be either
  - An appropriately formatted string,
  - The filename of an appropriately formatted file, or
  - An appropriately structured Python `dict` object.

In either case, the results will be deposited into `query_results.sqlite`.

5.1 Query Format

The input file (or string) must be in JSON format. It must consist of a single object with any or all of the following properties.

Property	Properties[1]	Type	Description
`name`	`require`	string	Name must be exactly this (case-insensitive).
	`contains`	string	Name must contain this string (case-insensitive).
	`regex`	string	Name must match this regular expression (case sensitive).
`gameId`		int or list of ints	gameId must be one of these.
`description`	`contains`	string	Description must contain this string (case-insensitive).
	`regex`	string	Description must match this regular expression (case sensitive).
	`query`	string	Results will be ordered in part by results of this query (case-insensitive, uses TF-IDF).
	`importance`	int	(See notes)[2]
`image`	`require`	boolean	Requires that the image URL be present or absent.[3]
	`prefer`	boolean	Prefers that the image URL be present or absent.[3]
	`importance`	number	(See notes)[2]
`publicationYear`	`exactly`	int	Game must be published in this year (or missing).
	`before`	int	Game must be published in or before this year (or missing).
	`after`	int	Game must be published in or after this year (or missing).
	`includeMissing`	boolean	Publication year must or must not be missing.
	`prefer`	string or number	If a string, must be "new" or "old", if a number the closer the year of publication is to that number the better.
	`preferKnown`	boolean	Prefers that the publication year be present or absent.[3]
	`importance`	number	(See notes)[2]

5.1 Query Format

`players`	`includes`	int	Must allow this number of players.
	`maxAtLeast`	int	Maximum number of players must be at least this.
	`minAtMost`	int	Minimum number of players must be at most this.
	`includeMinMissing`	boolean	Minimum number of players must be present or absent.[3]
	`includeMaxMissing`	boolean	Maximum number of players must be present or absent.[3]
	`prefer`	string or number	If a string, must be “high” or “low”, if a number the closer the number of players is to including that number the better.
	`preferKnown`	boolean	Prefers that the number of players be present or absent.[3]
	`importance`	number	(See notes)[2]
`playTime`	`atLeast`	number	Must take at least this long to play (in minutes).
	`atMost`	number	Must take at most this long to play (in minutes).
	`includeMinMissing`	boolean	Minimum play time must be present or absent.[3]
	`includeMaxMissing`	boolean	Maximum play time must be present or absent.[3]
	`prefer`	string or number	If a string, must be “high” or “low”, if a number the closer to the given play time the better.
	`preferKnown`	boolean	Prefers that the play time be present or absent.[3]
	`importance`	number	(See notes)[2]
`minAge`	`atLeast`	number	Minimum age must be at least this.
	`atMost`	number	Minimum age must be at most this.
	`includeMissing`	boolean	Minimum age must be present or absent.[3]
	`prefer`	string or number	if a string, must be “high” or “low”, if a number the closer the minimum age is to that number the better.
	`preferKnown`	boolean	Prefers that minimum age be present or absent.[3]
	`importance`	number	(See notes)[2]
`ratings`[4]	`minRated`	int	Number of people providing ratings must be at least this.
	`maxRated`	int	Number of people providing ratings must be at most this.
	`minRating`	number	Average rating must be at least this.
	`maxRating`	number	Average rating must be at most this.
	`prefer`	string or number	If a string, must be “high” or “low”, if a number the closer the average rating is to that number the better.
	`preferKnown`	boolean	Prefers that the average rating be present or absent.[3]
	`importance`	number	(See notes)[2]
`weights`	`minWeighted`	int	Number of people providing weights must be at least this.
	`maxWeighted`	int	Number of people providing weights must be at most this.
	`minWeight`	number	Average weight must be at least this.
	`maxWeight`	number	Average weight must be at most this.
	`prefer`	string or number	If a string, must be “high” or “low”, if a number the closer the average weight is to that number the better.
	`preferKnown`	boolean	Prefers that the average weight be present or absent.[3]
	`importance`	number	(See notes)[2]
`expansions`	`minExpansions`	int	Number of expansions must be at least this.
	`maxExpansions`	int	Number of expansions must be at most this.
	`prefer`	string or number	If a string, must be “high” or “low”, if a number the closer the number of expansions is to that number the better.
	`importance`	number	(See notes)[2]

[Link Type][5]	`require`	object	Object's properties are *[Link Value]*: *[boolean]*, where each such property's value requires that said property be present or absent.[3]
	`prefer`	object	Object's properties are *[Link Value]*: *[number]*, where each property's value is a relative weight placed on that property's presence (if positive) or absence (if negative).
	`totalImportance`	number	(See notes)[2]
`myRatings`[4]	`importance`	number	(See notes)[2]
	[gameId]	number	For each `gameId` (as a string) included as a property, its value should be this user's 1 to 10 ranking of the associated game.
`filename`		string	If you wish the results of this query to be stored somewhere other than `query_results.sqlite`, specify the filename here.

6 Query Results

The results of each query will be an SQLite file with a single table consisting of games which meet the query's requirements. Each game will also include a predicted rating (in the `prediction` column of the table) which will be between 1 and 10 (inclusive).

[1] Each property except for `gameId` and `filename` will have a value of type object, each of which must have at least one of the listed properties.

[2] Importance numbers are relative to each other and are used as coefficients in the linear combination of the specified predictive ratings. All importances default to 1 if not specified.

[3] When a property takes a boolean value, including `true` for that value means the associated condition *must be or is preferred to be true*, including `false` for that value means that the associated condition *must be or is preferred to be false*, and not including that property means that the associated condition *is not checked.*

[4] While the `ratings` property deals with the average ratings as an aggregate property, the `myRatings` property compares how this user rated specific games to how other users rated specific games and used Singular Value Decomposition to come up with predictions about this user's ratings.

[5] There are nine acceptable link types which can be specified independently of each other. To find out what they are as well as what values they may take, please see the `links` table in the database.