VIDEO GAME SEARCH ENGINE

Shukai Fan, Shujie Yang, Siyuan He

kennyfan, yshujie, hesy@umich.edu

University of Michigan - School of Information

Abstract

In this project we build a web-interface based search platform specific on game search with vague descriptions. In addition to basic search, we provide functions including static ranking, filter on dislike, pseudo relevance feedback, spell correction, and query suggestion.

Introduction

It's sometime annoying when game players exploring new games on game stores like Steam, because the search engine only allows searching on names and tags. If the user only have a vague idea of which kinds of games he want to explore, the existing search engine may cause inconvenience.

Here we present a game specific search engine allowing vague search on description. The search engine is equipped with other features like popular search engines to achieve higher accuracy and easier use.

Data Source

We obtain data from *Kaggle Steam Store Games* dataset [2]. The dataset has 27075 records, and each record represents a game with various attributes including appid, name. price, tags, description and so on. The information in the dataset is stored in separate files, so we need to pre-process the dataset before using it.

We first integrate different files by appid, and then drop the columns we don't use. The cleaned dataset contains 13 columns in total. including *appid, name, release date, price, genres, steam tags, positive ratings, negative ratings, owners, median playtime, detailed description, about the game*, and *short description*.

Features

BM25 based Ranking

The basic searching function is based on the BM25. BM25 is a kind of tf-idf-like retrieval functions with probabilistic retrieval framework [1].

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) * \frac{f(q_i,D) * (k_1+1)}{f(q_i,D) + k_1 * (1 - b + b * \frac{|D|}{\text{avgdl}})} \quad (1)$$

where $f(q_i, D)$ is the term frequency, and $\text{IDF}(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$ is the inverse document frequency. b and k_1 are constants.

In our project, we obtain scores from various documents, such as description, name and tag. The scores are added to form a score resemble with different weights .

Static Ranking

In order to take the quality of games into consideration and press penalty on low quality games, we introduce static ranking method in our platform. Static rank represents the value of each item and independent from the input query.

We calculate the static rank from the size, the popularity, the positive review rate, the total review number, the freshness and the median playtime, along with the grading scores from IGN websites. Static rank is combined with the BM25 score to decide the final rank.

Filter on Dislike

In the web interface we introduce a "dislike" button to provide feedback on the results. Once user click the button, the system will drop the disliked game from future results and re-calculate the results. In the re-calculation, the games close to the disliked one will be punished according to the similarity. In other word, if user dislike a game, the similar games will become less possible to appear.

Pseudo Relevance Feedback

Before show the results to user, we will re-run the progress for two extra feedback rounds. In this period, the top 10 results are considered as relevant while others will be considered as irrelevant. The top 10 results serve as the pseudo feedback of "like", so we will increase the rank of games similar to the top 10. When offering feedback for each of them, the query representation will be modified in the direction of high-ranked documents' representation (also in the reverse direction for irrelevant documents' representation). Moreover, normalization is provided with respect to the document number. The results after 2 extra rounds will be shown to the user as final result.

The idea is from *Rocchio algorithm*, i.e. most of documents within a class is near to a central point. By two rounds of pseudo relevance feedback, we can raise the rank of documents similar to the high-relevant results. As a result, modified query representation will be pull towards the center of the target game cluster.

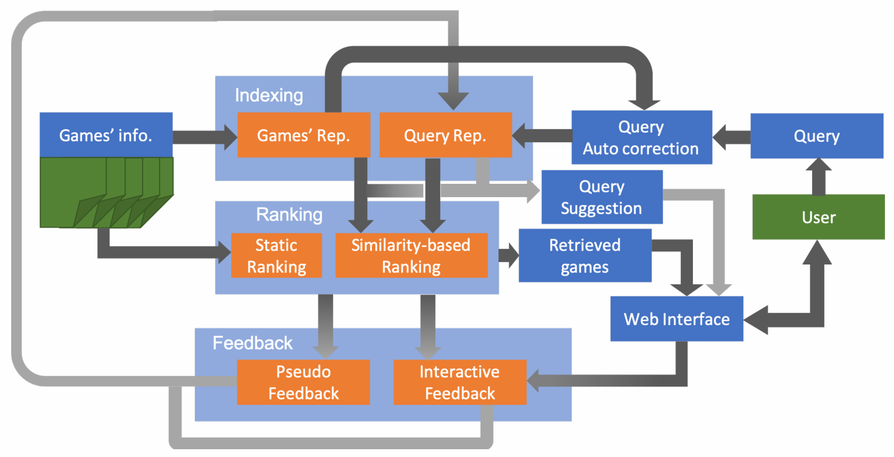

Figure 1: The Workflow of our Search Engine

Spell Correction

Automatic spell correction is offered in our search engine for the user input in the web interface. If we encounter a word which is not in our lexicon, we will call the spell correction and search the corrected word again. The correction is based on the mean edit distance. Both the original query and corrected query is shown on the top of the Web Interface

Query Suggestion

Query Suggestion is another function aimed to improve the user searching experience in our implementation. With the help of query suggestion, users are expected to find the target game with less number of search actions. With in our implementation, the game search engine analyze the query after spell correction, by utilizing the tf-idf matrix of retrieved games and suggesting those tokens with high co-occurrence possibility but low occurrence rate in total document pool.

Web Interface

We developed a web interface by Django for our user to access our search engine. The search result is shown in the output table, each row represents a game retrieved, and the columns are the information of the game. The last column should be the description, but we don't show it here due to space limit. The suggested queries will be listed above the output table, user can simply apply them by click on the suggested queries. If spelling mistake is recognized in user input, the modified result will appear above the suggested queries.

GameSearch

Your search - **anceint egipt** does not match any games, so it has been modified to **ancient egypt**

Are you trying to search ancient egypt adventure, ancient egypt puzzles, ancient egypt hidden, ancient egypt evil, ancient egypt secrets?

The output table

	Name	Release date	Price	Genres	Negative ratings	Positive ratings	Owners	Median playtime
Dislike	Assassin's Creed Origins	Oct. 26, 2017, midnight	49.99	Action;Adventure;RPG	6007	31644	1000000-2000000	3143
Dislike	Predynastic Egypt	Oct. 10, 2016, midnight	6.99	Indie;Simulation;Strategy	98	1048	20000-50000	0

Figure 2: Result Page

References

[1] Wikipedia. 2019. Okapi BM25. Retrieved from `https://en.wikipedia.org/wiki/Okapi_BM25`.

[2] Kaggle. 2019. Steam Store Games (Clean dataset). Retrieved from `https://www.kaggle.com/nikdavis/steam-store-games`