Vertical Steam Game Search Engine

Shukai Fan, Shujie Yang, Siyuan He
{kennyfan,yshujie,hesy}@umich.edu
University of Michigan, School of Information
Ann Arbor, MI, USA

ABSTRACT

With the rapid growth of video game market, various game search engines have appeared on platforms and online shops. However, existing search engines are mainly based on titles and tags, which make it hard for potential vague search. In this project we build a web-interface based search platform specific on game search with vague descriptions. We integrate popular features into our search engine in order to improve performance and user-friendliness, such as static ranking, filter on dislike, pseudo relevance feedback, spell correction and query suggestion. The performance of our result is evaluated by user feedback and compared with *Steam* platform.

KEYWORDS

search engine, video games, query suggestion, static ranking, personalising, web interface

1 INTRODUCTION

In the past few years, computer games become increasingly popular, resulting in the rapid growth of number of games and game media evaluations. Unlike in the years before, nowadays user can't look through all newly-published games and media evaluation reports to decide which one to buy, hence a search engine is quite important and useful. There are many successful online stores and medias provide a game search engine, one popular example of them is *Steam.* The search engine in *Steam* provides key word and label search, result ranking options and various filters[1]. However, the search engine might not perform well on some detailed description when user specific on some minor particulars of a game. For example, a user wants a forest adventure game with crocodile inside, and if he search "crocodile", the top-rank results will be warfare games.

A description-based search engine is needed to satisfy these situations, which means a search engine taking details in game description as long as titles and labels into consideration. The search engine should rank higher the games with some description similar to the query than traditional game search engines. In addition, the games with higher quality and popularity should rank higher because we assume user will more probably enjoy it.

In this project, we present a game search engine based on game description among games in *Steam* store. We combine static ranking and pseudo relevance feedback with BM25 score to achieve better performance. For easier usage, we provide a web interface with spelling check, dislike filter, and query suggestion integrated.

The evaluation of our game search engine is mainly rely on user feedback. We have designed a feedback form and a rubric to compare our search engine with *Steam* search.

2 WORKFLOW

The workflow of our vertical steam game search engine is depicted in Figure 1. In this section, we will walk through the general logic behind our product and detailed implementation is going to be covered in Section 3. The general process of our search engine can be divided into two part. One is user interactive part, which is marked in dark blue in Figure 1. Another is internal score based ranking part, which is marked in light blue in Figure 1.

In the user interactive part of this search engine, users' queries is considered as the primary input of the system and is pre-processed with query auto correction technique before be fed into the internal score based ranking part. The primary output displayed to the user is the list of retrieved games on the web interface, along with several query suggestions that users may potentially get interested. Users are also allowed to provide preference feedback towards the retrieval results within this interactive part. The feedback is also considered as a input towards the internal score based ranking part and the retrieval results is recalculated once the feedback is received.

For the internal score based ranking part inside our search engine, the ranking system is based on BM25 ranking, static ranking and pseudo feedback system. For BM25 ranking part, only the degree of text relevance between query and games' name/description/genres is consider in the ranking score. In static ranking part, every game has a invariable score considering the quality of the game. Given a query from user, the base score is a combination of a static score and a BM25 score. Before a ranked list of shown to the user, two rounds of pseudo feedback will be applied on the ranked list from base score, in order to promote other related but unmentioned keywords and refine the results.

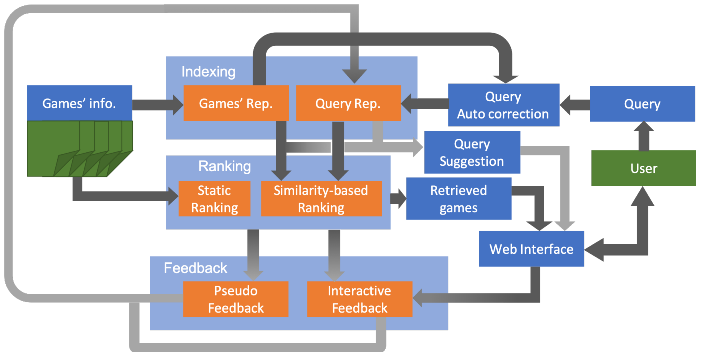


Figure 1: The Workflow of our Search Engine

3 METHODS

Relevance Scoring

In our vertical game search engine, the basic internal relevance ranking function is based on the BM25, calculating the relevance score between the query and the game document. BM25 is a kind of tf-idf-like retrieval functions with probabilistic retrieval framework [3].

$$S_{long}(D,Q) = \sum_{i=1}^{n} \mathrm{IDF}(q_i) * \mathrm{TF}(q_i,D) * \mathrm{QTF}(q_i,Q) \tag{1a}$$

$$\mathrm{TF}(q_i,D) = \frac{f(q_i,D)*(k_1+1)}{f(q_i,D)+k_1*(1-b+b*\frac{|D|}{\mathrm{avgdl}})} \tag{1b}$$

$$\mathrm{IDF}(q_i) = \log\frac{N-n(q_i)+0.5}{n(q_i)+0.5} \tag{1c}$$

$$\mathrm{QTF}(q_i,Q) = \frac{(k_3+1)*c(q_i,Q)}{k_3+c(q_i,Q)} \tag{1d}$$

where $f(q_i,D)$ is the term count of q_i in document D, and $c(q_i,Q)$ is the word count of q_i in the query Q. Here $\mathrm{TF}(q_i,D)$ is the normalized term frequency, $\mathrm{IDF}(q_i)$ is the pivoted invert document frequency, and $\mathrm{QTF}(q_i,Q)$ is smoothed equivalent query term frequency. b, k_1 and k_3 are constants. In our project, they are 0.75, 2 and 1000 respectively.

Beside relevance between game description and query, we put the name and tag into consideration as well. Due to the short text property of name and tag, only term frequency is valued, i.e.

$$S_{short}(D,Q) = \sum_{i=1}^{n} *\mathrm{TF}(q_i,D) * \mathrm{QTF}(q_i,Q). \tag{2}$$

Our game search engine is essentially a description based game search engine, therefore, the $S_{short}(D,Q)$ only makes up 10% of the final relevance score.

$$\begin{aligned} S(D,Q) = 0.9 &* \mathrm{MaxMinNorm}(S_{long}(D,Q)) \\ &+ 0.1 * \mathrm{MaxMinNorm}(S_{short}(D,Q)). \end{aligned} \tag{3}$$

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.

$$\begin{aligned} S(D,Q) = w_1 &* \mathrm{MaxMinNorm}(freshness) \\ &+ w_2 * \mathrm{MaxMinNorm}(pp) \\ &+ w_3 * \mathrm{MaxMinNorm}(log(r+1)) \\ &+ w_4 * \mathrm{MaxMinNorm}(log(npr+1)) \\ &+ w_5 * \mathrm{MaxMinNorm}(log(mp+1)) \\ &+ w_6 * \mathrm{MaxMinNorm}(IGNr) \end{aligned} \tag{4}$$

where f represent the freshness of the game (time after it is first published), pp represent the positive review percentage, r is the total number of reviews, npr is the number of net positive rating number, mp is the medium playtime of this game, $IGNr$ is the media review rate from IGN [1]

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.

[1]https://www.ign.com/

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.

$$\begin{aligned}\vec{Q_m} &= (a \cdot \vec{Q_o}) + (b \cdot \frac{1}{|D_r|} \cdot \Sigma_{\vec{D_j} \in D_r} \vec{D_j}) \\ &+ (c \cdot \frac{1}{|D_{nr}|} \cdot \Sigma_{\vec{D_k} \in D_{nr}} \vec{D_k}))+\end{aligned} \tag{5}$$

The idea is from *Rocchio algorithm*, i.e. most of documents within a class is near to a central point. Here D_r represents the document that are considered relevant, otherwise D_{nr}. In our implementation, only top 10 is considered relevant. 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.

Spell Correction

Automatic spell correction of the user input is provided in our search engine. If we encounter a word which is not in our corpus, we will call the spell correction function. This function will match the word to the most similar word in the corpus. The similarity is based on Levenshtein distance [2], also named minimum edit distance.

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) & \text{if min(i,j)=0} \\ min \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise} \end{cases} \tag{6}$$

Both the original query and corrected query will be shown on the top of our 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.

4 WEB INTERFACE

We developed a web interface using Django for our user to access our search engine. An example of search results is shown in Figure 2. In this example, the search query is "an1cien3t egypt". Note that we misspell the word "ancient" on purpose in order to test the spell correction function. From the line under the GameSearch button, we can see that our model identifies that this query does not match any items in our dataset, so it has been corrected to "ancient egypt". Below this line, some query suggestions are provided with a hyperlink in each suggested query to the search result of the corresponding query. In the output table, each row represents a game retrieved, and the columns are some basic information of the game including name, genres, price, owners, etc. The first column is the "Dislike" button, by clicking which users can filter out the corresponding game and some similar games that they don't want to see.

5 EVALUATION AND RESULTS

Evaluation Design

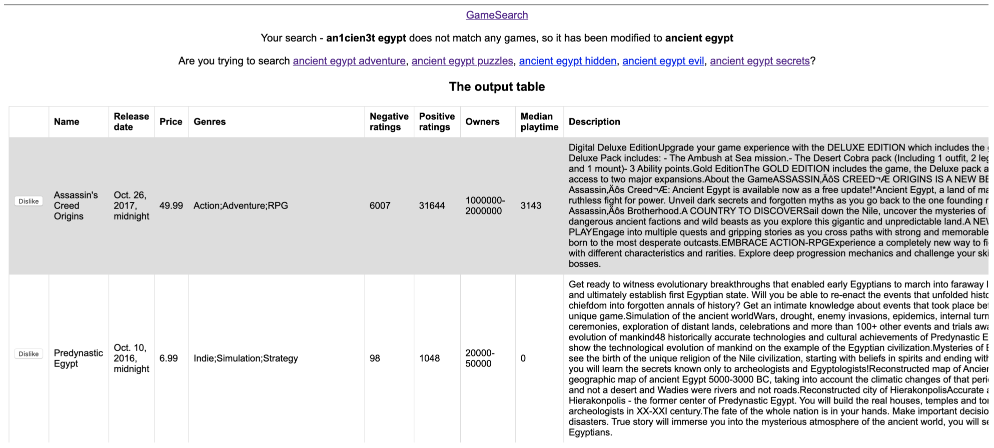
Since the performance of search engine is how well the user is satisfied, the evaluation of our game search engine will mainly based on the user feedback. The feedback is expected to provide evidence for evaluating the features added and the overall performance compared with existing systems. We have designed a feedback form and a series of actions to complete the feedback form.

Firstly, we ask the user to label the proportion of relevant documents to his query input. In order to check the effect of pseudo feedback and static rank, we ask the user to taking another round of attempt on the system without pseudo feedback or static rank using the same query.

Then we ask the user to think 5 games the want and some vague description of each game at first thinking. The user will conduct a series of search starting from the vague description, and we will record the search cost of each game, aka the number of search until find the target game. This evaluation will be repeated on the *Steam* as comparison.

During all the process above, we will record the proportion of relevant suggested queries to find out the performance of query suggestion.

The feedback form is attached in the appendix.



GameSearch

Your search - **an1cien3t egypt** 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	Description
Dislike	Assassin's Creed Origins	Oct. 26, 2017, midnight	49.99	Action;Adventure;RPG	6007	31644	1000000-2000000	3143	Digital Deluxe EditionUpgrade your game experience with the DELUXE EDITION which includes the Deluxe Pack includes: - The Ambush at Sea mission.- The Desert Cobra pack (Including 1 outfit, 2 le and 1 mount)- 3 Ability points.Gold EditionThe GOLD EDITION includes the game, the Deluxe pack a access to two major expansions.About the GameASSASSIN,ÄôS CREED¬Æ ORIGINS IS A NEW BE Assassin,Äôs Creed¬Æ: Ancient Egypt is available now as a free update!*Ancient Egypt, a land of ma ruthless fight for power. Unveil dark secrets and forgotten myths as you go back to the one founding Assassin,Äôs Brotherhood.A COUNTRY TO DISCOVERSail down the Nile, uncover the mysteries of dangerous ancient factions and wild beasts as you explore this gigantic and unpredictable land.A NE PLAYEngage into multiple quests and gripping stories as you cross paths with strong and memorable born to the most desperate outcasts.EMBRACE ACTION-RPGExperience a completely new way to f with different characteristics and rarities. Explore deep progression mechanics and challenge your sk bosses.
Dislike	Predynastic Egypt	Oct. 10, 2016, midnight	6.99	Indie;Simulation;Strategy	98	1048	20000-50000	0	Get ready to witness evolutionary breakthroughs that enabled early Egyptians to march into faraway and ultimately establish first Egyptian state. Will you be able to re-enact the events that unfolded histo chiefdom into forgotten annals of history? Get an intimate knowledge about events that took place be unique game.Simulation of the ancient worldWars, drought, enemy invasions, epidemics, internal turn ceremonies, exploration of distant lands, celebrations and more than 100+ other events and trials aw evolution of mankind48 historically accurate technologies and cultural achievements of Predynastic E show the technological evolution of mankind on the example of the Egyptian civilization.Mysteries of see the birth of the unique religion of the Nile civilization, starting with beliefs in spirits and ending wit you will learn the secrets known only to archeologists and Egyptologists!Reconstructed map of Ancie geographic map of ancient Egypt 5000-3000 BC, taking into account the climatic changes of that peri and not a desert and Wadies were rivers and not roads.Reconstructed city of HierakonpolisAccurate Hierakonpolis - the former center of Predynastic Egypt. You will build the real houses, temples and to archeologists in XX-XXI century.The fate of the whole nation is in your hands. Make important decisio disasters. True story will immerse you into the mysterious atmosphere of the ancient world, you will s Egyptians.

Figure 2: An example that is shown in our web interface: The input query is "an1cien3t egypt".

Evaluation Result

	Ours	Steam
Precision	0.85	0.91
Without Static Rank	0.83	/
Without Pseudo Feedback	0.79	/
Relevant Suggestion	0.33	/
Search Cost	2.91	3.14

Table 1: Evaluation Result

The mean of our feedback form collection is shown in table 1. From the precision view, aka the proportion of relevant results in top 10, *Steam* is better than our search engine. However, our search engine performs better in the search cost, aka the number of search needed to find a desired game. The evaluation shows that our game search engine is better in vague search, which means our system satisfies the design purpose.

In the backward elimination test, the evaluation score of the system without static rank or pseudo feedback is lower than the complete system, so we can claim the two units help enhance the performance. Note that the main effect of static rank is to adjust the ranking order and keep popular games higher, so the effect is not fully represented in the evaluation result data.

Discussion

One main problem in our evaluation is that the feedback is subjective. We can't ensure a user is using a consistent method to judge whether a game is relevant, which may cause the evaluation to be biased.

Meanwhile, we have only a limited number of samples and the sample variance on precision and search cost is large. This may be a potential problem that the sample can't reflect the actual result.

6 FUTURE WORK

Though our search engine reaches a high precision and performs well in human evaluation, some parts of it still need to be improved: (i) Currently, the spell correction part is based on Levenshtein distance [2], also called minimum edit distance. The time complexity is very high because it requires a comparison between the input word and every single word in the corpus, and then sort the (word, distance) tuple by distance to find the word that has the minimum edit distance to the input word. To increase the speed, we plan to implement another spell checker using hash table. (ii) By clicking the query suggestion item, another search that takes the query suggestion as input will be triggered. The model will start over and go through all the functions including static ranking, pseudo relevance feedback, and spell correction. However, it's not necessary to call all these functions or at least some of the functions can be modified to increase the time efficiency. So, our next step is to modify some unnecessary parts in the search process triggered by clicking query suggestion item. (iii) To further increase the precision, we will relate words or expressions in input query with some similar words or expressions. For example, "big" should be related to "large".

REFERENCES

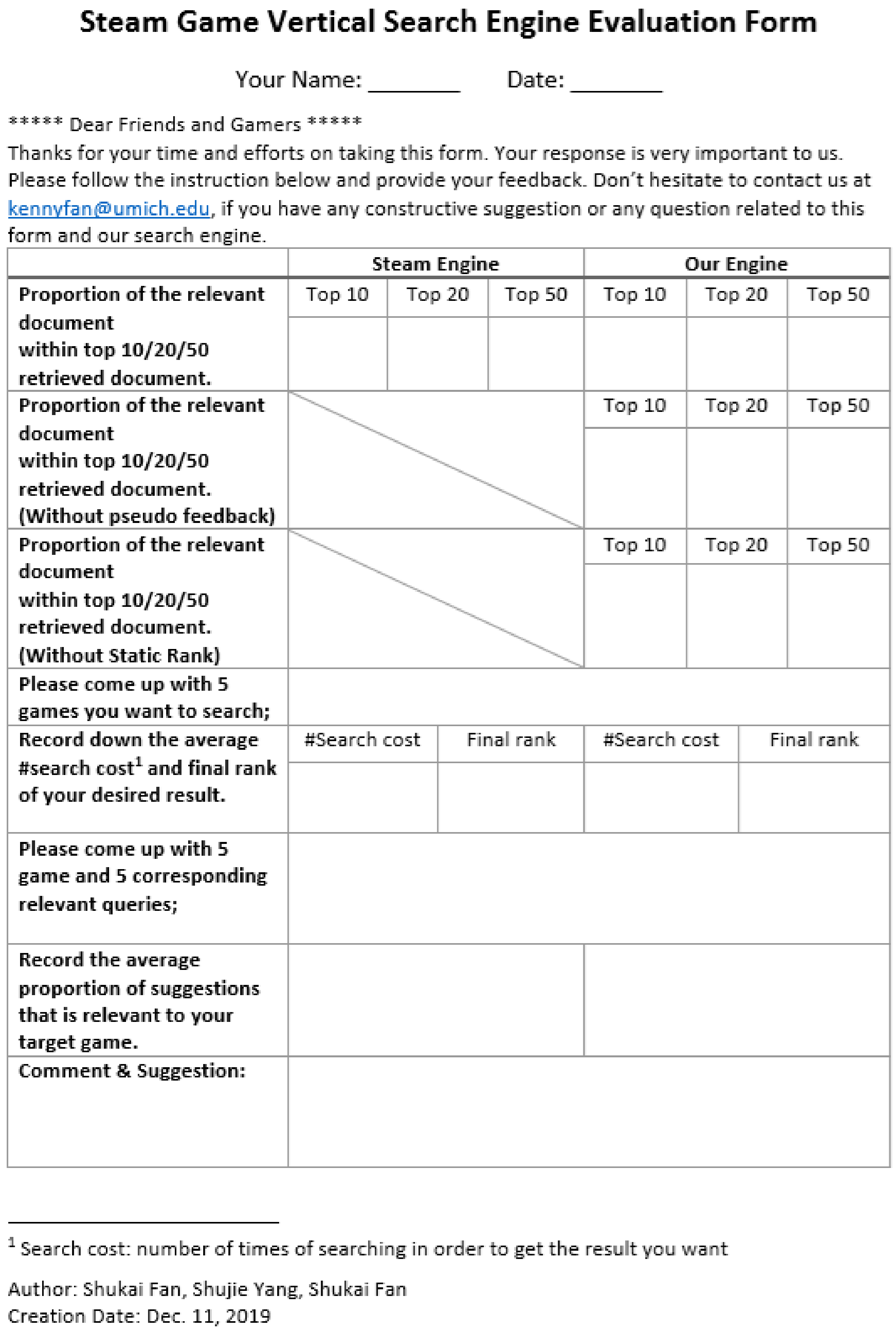
[1] Valve Corporation. [n.d.]. Steam Search. https://store.steampowered.com/search/. [Online; accessed 17-Dec-2019].

[2] Vladimir Iosifovich Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady* 10, 8 (feb 1966), 707–710. Doklady Akademii Nauk SSSR, V163 No4 845-848 1965.

[3] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3, 4 (April 2009), 333–389. https://doi.org/10.1561/1500000019

7 APPENDIX

User Feedback Form



Steam Game Vertical Search Engine Evaluation Form

Your Name: ________ Date: ________

***** Dear Friends and Gamers *****

Thanks for your time and efforts on taking this form. Your response is very important to us. Please follow the instruction below and provide your feedback. Don't hesitate to contact us at kennyfan@umich.edu, if you have any constructive suggestion or any question related to this form and our search engine.

	Steam Engine			**Our Engine**		
Proportion of the relevant document within top 10/20/50 retrieved document.	Top 10	Top 20	Top 50	Top 10	Top 20	Top 50
Proportion of the relevant document within top 10/20/50 retrieved document. (Without pseudo feedback)				Top 10	Top 20	Top 50
Proportion of the relevant document within top 10/20/50 retrieved document. (Without Static Rank)				Top 10	Top 20	Top 50
Please come up with 5 games you want to search;						
Record down the average #search cost[1] and final rank of your desired result.	#Search cost	Final rank		#Search cost	Final rank	
Please come up with 5 game and 5 corresponding relevant queries;						
Record the average proportion of suggestions that is relevant to your target game.						
Comment & Suggestion:						

[1] Search cost: number of times of searching in order to get the result you want

Author: Shukai Fan, Shujie Yang, Shukai Fan
Creation Date: Dec. 11, 2019