Information Processing Letters 115 (2015) 481–485

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Who wants a computer to be a millionaire?

Saeedeh Momtazi [*,1], Felix Naumann

Hasso Plattner Institute, Potsdam, Germany

ARTICLE INFO

Article history:
Received 15 November 2013
Received in revised form 1 December 2014
Accepted 10 December 2014
Available online 17 December 2014
Communicated by Jef Wijsen

Keywords:
Information retrieval
Question answering
Search engines
Knowledge bases

ABSTRACT

Competing against computers was one of the important challenges in the last decades. People like to compare their abilities with computers that are, in fact, their own invention. TV game shows provide a good opportunity for such competitions. Several attempts have been made to find out how sophisticated systems fare in game shows. An example of the task is competing in games with multiple-choice questions, such as *"Who wants to be a Millionaire"*.

We propose an approach to this problem by using search engines and knowledge bases to automatically select the answer. The experimental results indicate the superiority of the proposed model over related work. Our proposed method achieved average winnings of $250,000 on a US question set and became a millionaire six times, out of fifty runs, which is much higher than the normal winning rate among human contestants.

1. Computers and games

Games can be broadly categorized into the classes athletic, strategic, and knowledge-based games (and combinations). Human contestants in all three classes are under more and more pressure from machines: robots are increasingly adept in athletics, for instance the RoboCup soccer championships. In strategic games, such as chess or checkers, computers have made much more headway, for instance IBM's Deep Blue computer won over world champion Garri Kasparov in 1997 [3]. As more and more knowledge becomes available to computers, the possibility of competitively playing knowledge-based games is more and more apparent. Again an IBM QA system, Watson, famously competed in the game show Jeopardy, defeating the two best human contestants [1].

In this paper we focus on the game *"Who wants to be a Millionaire"*, which is a simpler task of the knowledge-based games. The game was first shown in the UK and has now spread to over 100 countries. Contestants are presented a succession of 15 increasingly difficult multiple-choice questions with four choices of which exactly one is correct. The prize money is increased with every correctly answered question, starting from £100, approximately doubling every round, and finally reaching £1,000,000. After reading each question and possible answers, contestants can choose to abort the game and collect the prize money amount so far reached. The game is complicated by additional rules, such as lifelines.[2]

To answer questions, our system relies on external sources, in particular search engines and structured knowledge bases. The key idea is to submit queries to these sources, analyze and score the results, and choose the most promising one. If the confidence measure of the system for all choices is zero, we choose to abort the game, earning the prize of the level so far reached.

* Principal corresponding author.
E-mail address: momtazi@ukp.informatik.tu-darmstadt.de (S. Momtazi).
[1] Present address: Ubiquitous Knowledge Processing Lab (UKP-DIPF), German Institute for Educational Research and Educational Information, Schlossstrasse 29, 60486 Frankfurt/Main, Germany. Tel.: +49 69 24708866.

[2] http://faculty.gvc.edu/swinzenburg/millionaire_study.html.

http://dx.doi.org/10.1016/j.ipl.2014.12.005
0020-0190/

2. Scoring strategies

There are various techniques to select an answer using information retrieval and knowledge processing methods. We first describe several approaches from the literature, and then propose two novel approaches for this task, namely *search snippet frequency* and *DBpedia spotlight*.

For all techniques used in previous studies and also our proposed search snippet frequency approach, we need to gather web data using a search engine. To this aim, we use the retrieval API of Google.

2.1. Strategies from literature

2.1.1. Counting strategy

The counting approach hypothesizes that, given a text corpus, the terms of a question and its correct answer co-occur more often than the question terms with a wrong answer. To determine the co-occurrence between question terms and answer terms, the following algorithm is proposed by Lam et al. [4]:

1. Remove stopwords from the question, resulting in a general query q.
2. Construct a specific query for each answer a, b, c, and d of the form $q_x = \langle \text{“x”}, \mathrm{q} \rangle$ with x as one of the four answer strings, resulting in four queries q_a to q_d.
3. Submit each query to a search engine and determine the number of search hits for each query: $H(a)$ to $H(d)$.
4. Choose the answer query that produced the *highest* number of search hits. For negated questions, the questions containing the word “not”, the query with the *lowest* number of search hits is selected instead, since negated questions are best answered based on inverted results.

2.1.2. Distance strategy

Since pure counting strategies consider only the number of retrieved pages, they are barely capable of taking the content of the search results into account. The distance score strategy assumes that question and answer tend to be in close proximity to each other within the qualified documents.

The first two steps are similar to the previous approach, but the strategy differs in the evaluation of the retrieved results [4]:

3. Submit queries q_a to q_d and for each query retrieve the top K web pages $d_{a,i}$ to $d_{d,i}$ $(1 \leq i \leq K)$.
4. Calculate distance scores $s_{a,i}$ to $s_{d,i}$ for the corresponding documents.
5. Determine the maximum scores for each answer: $s_x = \max_i(s_{x,i})$.
6. Choose answer with the maximum score s_x, or the minimum score for negated questions.

To calculate the *distance score* $s_{x,i}$ given a specific document $d_{x,i}$ we used the following approach proposed by Lam et al. [4].

- Tokenize the document $d_{x,i}$, the question q, and the answer x. Remove all non-alpha-numeric characters.
- Iterate over the words w of the document in order.
- If w is an answer word in x then search for question words q around the current word within a radius of r. The parameter r tunes the measure of proximity between answer and question tokens.
- Increase the total score of the document for each question word q in the radius of the answer word w by a normalized value that is inversely proportional to the distance between w and q. Hence, greater gaps between answer and question tokens are penalized.
- Return the average score per answer word by dividing the calculated score by the number of found answer words in the document. This helps to reduce the score of documents that have an imbalance between the number of answer words.

2.1.3. Keyword association strategy

The main shortcoming of the counting and distance strategies is that the rank of answer options depends on the frequency of the answer option on the Web in general: there might be more search results for an answer that is very common on the Web but is the wrong answer to the question. To solve the problem, Tonoike et al. [6] introduced two terminologies to develop the keyword association strategy: the *backward* and the *forward association*.

Backward association aims at solving the problem by normalizing the search frequency of the complete query by the search frequency of the answer option, resulting in a number in $[0, 1]$:

$$BA(q, x) = \frac{H(q \cup \{x\})}{H(x)} \tag{1}$$

where q is the set of question keywords and x is an answer option.

The *forward association* is also used to normalize scores. This normalization, however, is performed by the search frequency of the question keywords:

$$FA(q, x) = \frac{H(q \cup \{x\})}{H(q)} \tag{2}$$

The *keyword association strategy* chooses an answer option either with the highest forward association or the highest backward association based on a set of rules defined by Tonoike et al. [6].[3] Again, the result is reversed for negated questions.

2.2. Search snippet frequency

Although the distance and keyword association approaches try to overcome the shortcomings of the counting techniques, they do not always perform well. We believe that using answer choices in the query is one of the main problems in all state-of-the-art approaches. Directly utilizing the answer options in the queries causes a bias toward answers and gives a higher priority to them compared

[3] Details of the rules can be found in their paper.