Artificial Intelligence 222 (2015) 157–181

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Playing with knowledge: A virtual player for “Who Wants to Be a Millionaire?” that leverages question answering techniques

Piero Molino, Pasquale Lops *, Giovanni Semeraro, Marco de Gemmis *, Pierpaolo Basile

Department of Computer Science, University of Bari Aldo Moro, Via E. Orabona 4, I-70125 Bari, Italy

A R T I C L E   I N F O

Article history:
Received 5 November 2013
Received in revised form 29 January 2015
Accepted 6 February 2015
Available online 12 February 2015

Keywords:
Language game
Question answering
Natural language processing
Artificial intelligence
Decision making

A B S T R A C T

This paper describes the techniques used to build a virtual player for the popular TV game “Who Wants to Be a Millionaire?”. The player must answer a series of multiple-choice questions posed in natural language by selecting the correct answer among four different choices. The architecture of the virtual player consists of 1) a *Question Answering* (QA) module, which leverages Wikipedia and DBpedia datasources to retrieve the most relevant passages of text useful to identify the correct answer to a question, 2) an *Answer Scoring* (AS) module, which assigns a score to each candidate answer according to different criteria based on the passages of text retrieved by the Question Answering module, and 3) a *Decision Making* (DM) module, which chooses the strategy for playing the game according to specific rules as well as to the scores assigned to the candidate answers.

We have evaluated both the accuracy of the virtual player to correctly answer to questions of the game, and its ability to play real games in order to earn money. The experiments have been carried out on questions coming from the official Italian and English boardgames. The average accuracy of the virtual player for Italian is 79.64%, which is significantly better than the performance of human players, which is equal to 51.33%. The average accuracy of the virtual player for English is 76.41%. The comparison with human players is not carried out for English since, playing successfully the game heavily depends on the players’ knowledge about popular culture, and in this experiment we have only involved a sample of Italian players. As regards the ability to play real games, which involves the definition of a proper strategy for the usage of lifelines in order to decide whether to answer to a question even in a condition of uncertainty or to retire from the game by taking the earned money, the virtual player earns € 114,531 on average for Italian, and € 88,878 for English, which exceeds the average amount earned by the human players to a greater extent (€ 5926 for Italian).

* Corresponding authors.
E-mail addresses: piero.molino@uniba.it (P. Molino), pasquale.lops@uniba.it (P. Lops), giovanni.semeraro@uniba.it (G. Semeraro), marco.degemmis@uniba.it (M. de Gemmis), pierpaolo.basile@uniba.it (P. Basile).

http://dx.doi.org/10.1016/j.artint.2015.02.003
0004-3702/© 2015 Elsevier B.V. All rights reserved.

1. Introduction

The work on intelligent computer games has a long history and has been one of the most successful and visible results of Artificial Intelligence research [34]. Indeed, today artificial systems are able to compete and sometimes challenge human players in several complex games. Most of these games are *closed world* ones, meaning that they have a finite number of possible choices, which allows the researchers to solve them in a formal way, even though they are hard to play due to the exponential dimensions of the search spaces. A more challenging type of games is represented by *open world* games, such as sport games or crosswords: they are less structured and, moreover, both the states of the game and the actions of the player cannot be easily enumerated, making the search through the space of possible solutions practically unfeasible. One of the most recent results in this field is the success of *Watson*, the open-domain question answering system built by IBM Research, which in February 2011 beat the two highest ranked players of the quiz show Jeopardy! [17,18].

We are particularly interested to games related to human language. They are classified in *word games*, in which word meanings are not important, and *language games*, in which word meanings play an important role [27]. Language games generally require a wide linguistic and common sense knowledge. "Who Wants to Be a Millionaire?" (WWBM) is a perfect example of a language game in which the player provides an answer to a question posed in natural language by selecting the correct answer out of four possible ones. Even though the number of possible answers is limited to four, being able to successfully play this game heavily depends on the player's knowledge, her understanding of the questions and her ability to balance the confidence in the answer against the risk taken in answering.

This article describes the architecture of a *Virtual Player* for the WWBM game, which leverages Question Answering (QA) techniques and both Wikipedia and DBpedia open knowledge sources in order to incorporate the knowledge useful for playing the game. A preliminary work that describes the architecture of the virtual player is presented in [37]. The current work extends the previous work along the following directions: the use of DBpedia; the *decision making* strategy integrated to manage the "lifelines" characterizing the game; the possibility to retire from the game; the use of machine learning techniques to improve the process of scoring the candidate answers to a question. Extended related work about question answering, answer validation and language games are also provided, along with more extensive experiments on both the Italian and the English versions of the game.

Motivated by the challenge to develop an effective virtual player for the WWBM game, in this paper we address two issues, one related to the more general topic of designing effective QA systems, while the other concerns specific aspects of the game. Hence, we investigate the following research questions:

- *RQ1. To what extent can a QA system be designed in a language-independent way, by preserving its effectiveness?*
  We cope with this question by proposing a general architecture of a QA and Answer Scoring (AS) framework which exploits resources or algorithms specifically designed for a given language exclusively for basic NLP operations, such as part-of-speech tagging or stemming. In order to assess the effectiveness of the framework for at least two different languages, we performed experiments on English and Italian.
- *RQ2. Can Wikipedia and DBpedia serve as effective knowledge bases for answering WWBM game questions?*
  We address this question by developing a virtual player based on the proposed QA framework, which leverages Wikipedia and DBpedia open knowledge sources to find the correct answers. Besides the QA framework, the virtual player adopts a decision making strategy to play the game with all its rules, i.e. usage of "lifelines", answering in a condition of uncertainty, retiring from the game by taking the earned money. Experiments are performed to compare the accuracy of the virtual player against that of human players.

The paper is organized as follows: Section 2 describes the rules of the game, while related work in the areas of language games, QA and AS are presented in Section 3. The architecture of the virtual player, the details of the QA and AS modules, and the decision making strategy adopted to play the game are provided in Sections 4–7. Section 8 reports the results of an extensive evaluation performed on Italian and English versions of the game. Finally, conclusions are reported in Section 9.

2. Rules of the game

WWBM is a language game, broadcast by many TV channels in several countries, in which a player must correctly answer a series of 15 multiple-choice questions of increasing difficulty. Questions are posed in natural language and the correct answer is selected among four possible choices.

Fig. 1 shows an example of the question *Who directed Blade Runner?*, and the four possible answers **A)** Harrison Ford **B)** Ridley Scott **C)** Philip Dick **D)** James Cameron. There are no time limits to answer the questions. Moreover, contestants read the question in advance, and then at any time they can decide whether to attempt an answer or quit the game by keeping the earned money. Each question has a certain monetary value (level 1: € 500; level 2: € 1000; level 3: € 1500; level 4: € 2000; level 5: € 3000; level 6: € 5000; level 7: € 7000; level 8: € 10,000; level 9: € 15,000; level 10: € 20,000; level 11: € 30,000; level 12: € 70,000; level 13: € 150,000; level 14: € 300,000; level 15: € 1,000,000). If the answer is correct, the player earns a certain amount of money and continues to play by answering questions of increasing difficulty until either she reaches the last question or she retires from the game by taking the earned money. There are three *guarantee points* where the money is banked and cannot be lost even if the player gives an incorrect answer to one of

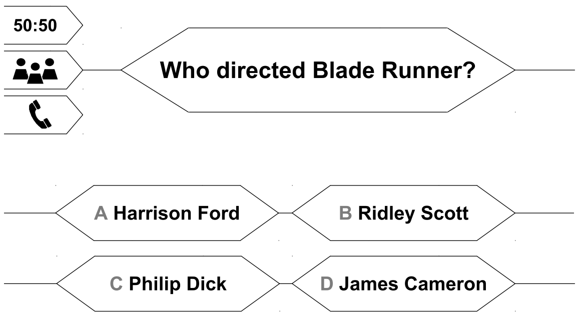


Fig. 1. An example of "Who Wants to Be a Millionaire?" question.

the next questions: 3000, 20,000 and 1,000,000 Euros, corresponding to the milestone questions 5, 10, 15, respectively. At any point, the contestant may use one or more of three "lifelines", which provide her with some form of assistance:

- *50:50*: this lifeline removes two wrong answers, leaving the player with a binary choice between the correct answer and the incorrect one;
- *Poll the Audience*: the player asks the studio audience to pronounce about the correct answer. The percentages of the audience for the 4 different answers are given to the player, who has the last word on the choice of the answer;
- *Phone a Friend*: the player has 60 seconds to phone a friend, and read the question with the four possible choices, in order to get a suggestion about the right choice.

3. Related work

3.1. Natural Language Processing and language games

Language games usually require a big amount of knowledge and deep reasoning capabilities to compete at human level. Artificial players for language games adopt Natural Language Processing (NLP) technologies in order to manage the complexity and ambiguity of the language, while storing and manipulating complex representations of the knowledge involved in the game.

A popular language game is solving crossword puzzles. Besides the linguistic knowledge, solving crosswords requires the satisfaction of constraints over the possible answers. The first experience reported in literature is *Proverb* [28], that exploits large libraries of clues and solutions to past crossword puzzles, while WebCrow [15], the first solver for Italian crosswords, exploits the Web as the main source of information, and a set of previously solved games, as well. WebCrow is based on the sequential combination of "clue answering" and "grid filling", a solution which is radically different from a human approach. In order to find the best candidate words, WebCrow queries Google search engine with queries that are reformulations of the original definitions obtained by enriching the morphological forms of the keywords (e.g., by varying number and gender for nouns, or the tense for verbs), by adding synonyms and hypernyms from WordNet, in order to make the querying process more effective. The text in the retrieved pages is analyzed by NLP techniques, and a classifier chooses the most probable part-of-speech depending on the definition in order to reduce the number of candidate words. Finally, a list of the best words for the definition is given and they are matched against the letter constraints given by the grid. WebCrow achieves 68.8% of correct words and 79.9% of correct letters, showing the potential of the Web as a resource for complex language games.

Another interesting language game is the Guillotine, a game broadcast by the Italian National TV company. It involves a single player, who is given a set of five words (clues), each linked in some way to a specific word that represents the unique solution of the game. Words are unrelated to each other, but each of them is strongly related to the word representing the solution. For example, given the five words *sin, Newton, doctor, pie, New York*, the solution is *apple* because: the apple is the symbol of original *sin* in Christian theology; *Newton* discovered the gravity by means of an apple; "an apple a day keeps the *doctor* away" is a famous proverb; the apple *pie* is a fruit pie, and *New York* city is also called "the big apple". In [48,3], the authors present OTTHO (On the Tip of my THOught), an artificial player for the Guillotine game. The idea behind OTTHO is to define a *knowledge infusion* process which analyzes unstructured information stored in open knowledge sources on the Web to create a memory of linguistic competencies and world facts that can be effectively exploited by the system for a deeper understanding of the information it deals with. The knowledge infusion process adopts NLP techniques to build a knowledge base and, similarly to the approach described in this article, extracts information mainly from Wikipedia. A reasoning mechanism based on a spreading activation algorithm is adopted to retrieve the most appropriate pieces of knowledge useful to find possible solutions.

An approach to implement a virtual player for the "Who Wants to Be a Millionaire?" game has already been proposed [25]. The authors exploit the huge amount of knowledge in the Web and use NLP techniques to reformulate the questions in order to create different queries. The queries are then sent to Google search engine and the number of results is used as a ranking mechanism which exploits the redundancy of the information sources [25]. A decision making module, that combines results in the spirit of ensemble learning using an adaptive weighting scheme, tries to maximize the earned amount of money with respect to the risk of answering. The system reaches an accuracy of 75%, showing how unstructured data can be useful for this kind of task, but it fails when questions require common sense reasoning and access to structured information. The main differences with respect to our work are that we adopt selected sources of information available on the Web, such as Wikipedia and DBpedia, rather than the whole Web, in an attempt to improve reliability of the answers; moreover, we adopt a QA framework instead of a search engine in order to improve the process of selecting the most reliable passages.

In February 2011 the IBM Watson supercomputer, adopting technology from the DeepQA project [18], has beaten two champions of the Jeopardy! TV quiz. In Jeopardy! the player is given a question but expressed as answer, and has to find the answer which must be expressed as a question. Watson applies several different NLP, Information Retrieval (IR) and Machine Learning (ML) techniques focusing on open-domain QA, by answering questions without domain constraints. Watson analyzed 200 millions content elements, both structured and unstructured, including the full text of Wikipedia. The steps of the Watson answering process can be summarized as follows: 1) it acquires knowledge from different data sources, namely encyclopedias, dictionaries, thesauri, journal articles, databases, taxonomies and ontologies; 2) the input of the problem is treated as a question, then it is analyzed using NLP algorithms and lastly it is classified; 3) candidate answers are generated from the previously acquired knowledge; 4) candidate answers that do not pass a threshold are filtered out. Several scoring algorithms are used to rank the candidate answers in order to give evidence of their quality. Lastly, the scoring criteria are combined to select the final candidate answer. Similarly to the approach implemented in Watson, we adopt QA techniques for solving the WWBM game, and we use the same process of using different scoring criteria, which are eventually combined to return the best candidate answer.

3.2. Question answering

The task of Question Answering is to find correct answers to users' questions expressed in natural language. The traditional QA pipeline relies on the following steps: Question Analysis, Passage Retrieval, Answer Extraction, Answer Selection/Validation. *Closed-domain QA* refers to QA tasks pertaining specific and limited domains (such as medicine). Dealing with questions in a closed-domain is generally an easier task since some kind of domain-specific knowledge can be exploited, and only a limited type of questions are accepted. *Open-domain QA* does not refer to a specific domain and deals with more general questions. This usually requires the use of world knowledge from which the answer is extracted. The Web is generally used as source of knowledge in order to exploit the redundancy of information, by selecting the answers according to their frequency among the search results [14,26]. This technique is often complemented by textual pattern extraction and matching in order to find the exact answers and rank them by confidence [23,39]. NLP methods are used for understanding users' questions and matching passages extracted from the documents [22,24]. As reported in [10], the adoption of NLP plays a key role, since there are likely too few answers to users' questions, and the way in which they are expressed may significantly differ from the question. Thus, NLP is essential for discovering complex lexical, syntactic and semantic relationships between questions and candidate answers. The most commonly adopted linguistic analysis steps include stopword removal, stemming, lemmatization, part-of-speech tagging, parsing, named entity recognition, word sense disambiguation and semantic role labeling. Other approaches rely on a different kind of knowledge to extract the answers to questions. In [16], an approach to open-domain QA over massive knowledge bases (KBs) is carried out by decomposing the full open QA problem into smaller sub-problems including question paraphrasing and query reformulation. In the first step, the question is rewritten using a paraphrase operator, mined from a large corpus of questions, in order to reduce the variance of the input questions: for example "How can you tell if you have the flu?" is reformulated as "What are signs of the flu?". The second step uses hand-written templates to parse the paraphrased question to a specific KB query, which is then reformulated through a query-rewrite operator to cope with the vocabulary mismatch between question words and KB symbols.

In our work we adopt Wikipedia and DBpedia knowledge sources to extract answers to questions, and we integrate common linguistic analysis steps. The novelty is the adoption of Distributional Semantic Models (DSM) [47] for computing the semantic similarity between questions and documents. This is novel in the QA field, especially for the task of answer re-ranking.

In [4] the authors propose a semantic parser that maps questions to answers via latent logical forms. They first generate a number of logical forms exploiting a lexicon that maps logical predicates to phrases built from a knowledge base and a large text corpus. They then calculate the probability of each candidate logical form with a log.linear model that exploits lexical and logical features, trained on question-answer pairs. This approach differs from ours because we rely mainly on lexical features for retrieving passages rather than mapping questions to their logical form, and we try to map questions to DBpedia triples exploiting predicate lexicalization, which has some commonalities with the lexicon construction step.

3.3. Question answering for machine reading and answer validation

QA systems generally deal with simple questions that require almost no inference to find the correct answers, since no real understanding of documents is performed. This historically led to QA architectures based on Information Retrieval techniques, in which the final answers are obtained after focusing on selected portions of retrieved documents and matching sentence fragments or sentence parse trees. Other systems perform a deeper analysis of texts, to solve tasks that involve some kind of reasoning. For example, in the Machine Reading task [40], the goal is to answer questions that require a deep knowledge of individual short texts and in which systems are required to choose one answer, by analyzing the corresponding test document in conjunction with background text collections. Other complex tasks include Recognizing Textual Entailment (RTE) [12][1] and the Answer Validation Exercise (AVE) [45].[2] In RTE, a system must decide whether the meaning of a text T entails the meaning of a different text H, i.e. the hypothesis. Differently, AVE consists in deciding whether an *answer* to a *question* is correct or not according to a *given text*. Systems receive a set of triplets (question, answer and supporting text) and must return a value for each triplet, which can be: VALIDATED, i.e. the answer is correct and supported, although not selected, SELECTED, i.e. the answer is validated and chosen as the output, and REJECTED, i.e. the answer is incorrect or there is no enough evidence of its correctness. In order to solve these tasks, several techniques were adopted, mostly based on the use of lexical processing, syntactic processing, and Named Entities [44].

In [6] the answer validation is carried out by computing the overlap between the system response to a question and the stemmed content words of a human-generated answer key. The intuition behind this strategy is that a good answer is expected to contain certain keywords, but the exact phrasing does not matter. This is the reason why the authors used stopwords removal and stemming. In [32], the answer validation is based on the intuition that the knowledge which connects an answer to a question can be estimated by exploiting the redundancy of the Web information. More specifically, the hypothesis is that the number of documents retrieved from the Web in which the question and the answer co-occur is a good indicator of the validity of the answer.

More complex strategies propose solutions to answer validation based on a lightweight process of abduction starting from paragraphs of text where the answers can be found [21], or based on the use of semantics. For example, Castillo [9] builds a model using Support Vector Machines to define whether the implication holds, using a set of lexical and semantic measures to compute the similarity between (hypothesis, text) pairs. Features are based on: 1) the overlap between text and hypothesis (computed by taking into account single words or stems, bigrams and trigrams); 2) the cosine similarity and the Levenshtein distance between the text and hypothesis; 3) the semantic similarity using Wordnet. Glöckner [20] also used shallow feature extraction (like lexical overlap) to validate answers. A local score is then computed, and aggregation is used to determine a combined score for each answer which captures the joint evidence of all snippets supporting the answer. Ahn et al. [2] exploited semantic representations and inference techniques in the process of answer extraction from selected passages. Answer candidates are extracted using a "relaxed" unification method that takes advantage of Prolog unification, which allows to assign high scores to perfect matches between terms of the question and passage, and low scores for less perfect matches obtained by relaxed unification. Less perfect matches are granted for different semantic types, predicates with different argument order, or terms with symbols that are semantically related (hypernymy) according to WordNet.

We were inspired by those previous approaches to build the virtual player for the WWBM game, and we borrowed most of the techniques adopted in the AVE task, such as techniques based on lexical processing and computation of an approximate match between passages of text and candidate answers (Section 6).

3.4. Question answering over linked data

The rapid growth of semantic information published on the Web, in particular through the linked data initiative [5], poses new challenges when supporting users to query these large amounts of heterogeneous and structured semantic data using natural language interfaces. This led to the rise of ontology-based QA, a new paradigm able to exploit the expressive power of ontologies and to go beyond the representation of user information needs as keyword-based queries.

FREyA [13] allows users to enter queries in any form. In a first step, it generates a syntactic parse tree in order to identify the answer type. The processing then starts with a lookup, annotating query terms with ontology concepts using the ontology-based gazetteer OntoRoot. Next, on the basis of the ontological mappings, triples are produced and finally combined to generate a SPARQL query. In a similar way, PowerAqua [30] transforms the query into a set of triples ⟨SUBJECT, PROPERTY, OBJECT⟩ by means of linguistic processing, and maps the triples to suitable semantic resources in various ontologies that are likely to describe the query terms. Given these semantic resources, a set of ontology triples that jointly cover the query is derived and combined into a complete answer, by merging and ranking the various interpretations produced in different ontologies.

QAKiS [8] is a QA system over DBpedia that focuses on bridging the gap between natural language expressions and labels of ontology concepts by means of the WikiFramework repository. This repository has been built by automatically extracting relational patterns from Wikipedia free text that specify possible lexicalizations of properties in DBpedia. For example, one of the natural language patterns that express the relation BIRTHDATE is WAS BORN ON. The approach of using a pattern repository represents a promising solution for bridging the lexical gap between natural language expressions and ontology labels, which we also used in our work (Section 5.1). Similarly to QAKiS, SemSek [1] also focuses on matching natural language expressions to ontology concepts. This relies on three steps: linguistic analysis, query annotation, and

[1] http://pascallin.ecs.soton.ac.uk/Challenges/RTE/.
[2] http://nlp.uned.es/clef-qa/ave/.

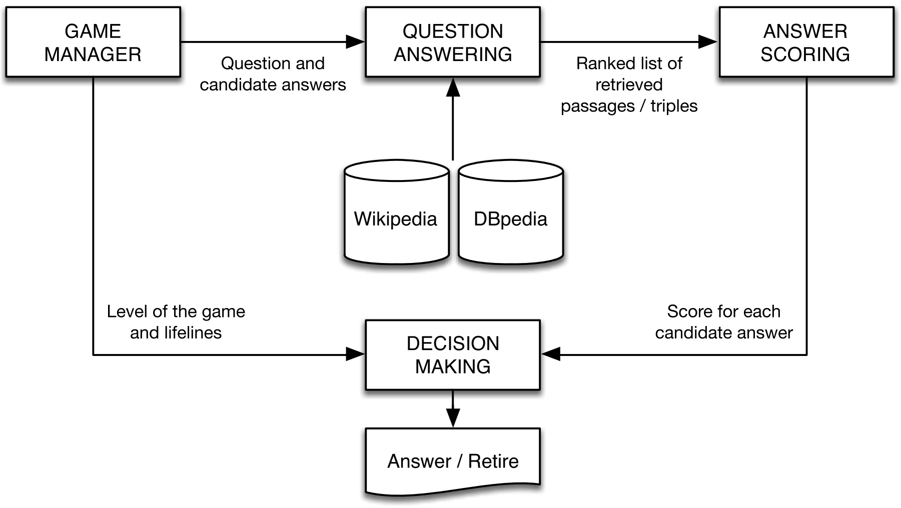

GAME MANAGER
Question and candidate answers
QUESTION ANSWERING
Ranked list of retrieved passages / triples
ANSWER SCORING
Wikipedia
DBpedia
Level of the game and lifelines
Score for each candidate answer
DECISION MAKING
Answer / Retire

Fig. 2. Virtual player architecture.

semantic similarity. The query annotation mainly looks for the entities and classes in a DBpedia index that match the expressions occurring in the natural language question. This process is guided by the syntactic parse tree provided by the linguistic analysis. Starting from the most plausible identified resources and classes, SemSek retrieves an ordered list of terms following the dependency tree. In order to match these terms to DBpedia concepts, SemSek uses two semantic similarity measures, one based on Explicit Semantic Analysis [19], and one based on WordNet.

In [31], the evaluation of these systems in the context of the challenges for question answering systems over linked data (QALD) is presented. Results are encouraging and show that QA over linked data can deliver answers to quite complex information needs expressed in natural language using heterogeneous semantic data, even though the task of mapping natural language to formal queries is not trivial and still has several problems [11].

4. Virtual player architecture

The architecture of the virtual player for the WWBM game is presented in Fig. 2 and consists of four modules:

- Game Manager: manages the user interface, selects a question for each level of the game, and logs the information about each game for the different players;
- Question Answering: leverages Wikipedia and DBpedia to retrieve and rank the most relevant passages of text useful to identify the correct answer to a question. More details are provided in Section 5;
- Answer Scoring: leverages the list of text passages extracted from the Question Answering module and adopts several criteria to assign a score to each of the four possible answers to a question. A detailed description is provided in Section 6;
- Decision Making: takes the final decision whether answering to a question or retiring from the game by considering the current level of the game, the available lifelines, and the score computed for each possible answer by the Answer Scoring. A detailed description is provided in Section 7.

According to the current level of difficulty of the game, the Game Manager selects a question along with the four possible answers, which are then passed to the Question Answering module. This module exploits the knowledge contained in Wikipedia and DBpedia to select a ranked list of passages of text that likely contain the correct answer to the question. These passages are processed by the Answer Scoring, which implements a set of heuristics to come up with a score for each possible answer to the question. Finally, the Decision Making module decides to provide a specific answer or to retire from the game, by taking into account the scores of the candidate answers, the available lifelines and the current level of the game.

In order to better explain the whole process of answer selection, we will use the following running example throughout the paper. Let us consider the question in Fig. 1: *Who directed Blade Runner?*, whose correct answer is B) Ridley Scott. Let us also consider the ranked list of text passages provided by the Question Answering module in Table 1.

Each passage contains the title of the Wikipedia page from which it was extracted, and the score computed by the Question Answering module. More formally, given $\langle q, (A, B, C, D)\rangle$, where q is the question and (A, B, C, D) are the four possible answers, the Question Answering module returns a list of results $R_q = \{\langle t_1, p_1, w_1\rangle, \ldots, \langle t_{|R_q|}, p_{|R_q|}, w_{|R_q|}\rangle\}$ where the triple $\langle t_i, p_i, w_i\rangle$ corresponds to the title t_i of the Wikipedia page containing the passage p_i, and w_i is the score

Table 1
List of passages returned by the Question Answering module for the question "*Who directed Blade Runner?*". Each passage reports the Wikipedia page it is extracted from and the relevance score of that passage with respect to the question.

Wikipedia page (t_i)	Passage (p_i)	QA score (w_i)
Ridley Scott	Sir Ridley Scott (born 30 November 1937) is an English film director and producer. Following his commercial breakthrough with Alien (1979), his best-known works are the sci-fi classic Blade Runner (1982) and the best picture Oscar-winner Gladiator (2000).	5.32
Blade Runner	Blade Runner is a 1982 American dystopian science fiction action film directed by Ridley Scott and starring Harrison Ford, Rutger Hauer, and Sean Young. The screenplay, written by Hampton Fancher and David Peoples, is loosely based on the novel Do Androids Dream of Electric Sheep? by Philip K. Dick.	5.1
Blade Runner	Director Ridley Scott and the film's producers "spent months" meeting and discussing the role with Dustin Hoffman, who eventually departed over differences in vision. Harrison Ford was ultimately chosen for several reasons.	5
Blade Runner	The screenplay by Hampton Fancher was optioned in 1977. Producer Michael Deeley became interested in Fancher's draft and convinced director Ridley Scott to film it.	4.9
Blade Runner	Interest in adapting Philip K. Dick's novel Do Androids Dream of Electric Sheep? developed shortly after its 1968 publication. Director Martin Scorsese was interested in filming the novel, but never optioned it.	1.2

indicating the relevance of that passage with respect to the question q. The list R_q is empty if the Question Answering module is not able to find passages relevant to the question. This may happen when:

- the information is not contained in any of the adopted knowledge sources, i.e. Wikipedia or DBpedia;
- the information is contained in one of the adopted knowledge sources, but it is not possible to find a match (textual or semantic) between the question and the passages containing the answer. For example, even though the answer to the question *When was Leonardo da Vinci born?* is contained in Wikipedia, it is difficult to find it, since it is reported as "Leonardo da Vinci (April 15, 1452–May 2, 1519)". We solve this problem by implementing a strategy based on the use of DBpedia (Section 5.1);
- the question falls into one of the categories which remain unanswerable by our system (Section 8.1.1).

5. Question answering

In order to provide the virtual player with the knowledge useful for finding the correct answers to the questions of the game, we exploited data coming from two open knowledge sources: Wikipedia and DBpedia. Their knowledge is processed by a multilingual QA framework, called QuestionCube [35,36], able to retrieve the most relevant passages of text which likely contain the correct answer to the questions of the game.

Fig. 3 depicts a high level architecture of the QA framework. It works in two separate steps: at indexing time the system builds two different indexes for documents and passages belonging to each document, while at query time the question is analyzed by an NLP pipeline, tagged with linguistic annotations, and passed to a set of search engines. Finally, the passages belonging to the documents retrieved by the search engines are scored, filtered and returned as a ranked list. More details follow:

Question analysis: a pipeline of NLP analyzers is adopted to tag questions with linguistic annotations. We adopt the following pipeline: *stemming*, *part-of-speech tagging*, *lemmatization*, and *Named Entity Recognition (NER)*. This representation is the input to the search engines and also to each filter, both of which need this information to carry out the analysis at the same linguistic level of the question. So far, NLP analyzers for English and Italian have been developed.

Search engines: the use of multiple engines allows the integration of several retrieval strategies and data sources. Each search engine has its own query generation component, since it may exploit different tags added to the query and may adopt a different query syntax. When a new question comes, each engine is activated and the lists of retrieved documents are merged into a single list, by maintaining the reference to the provenance of each document and the corresponding score computed by each engine. Starting from the retrieved documents, the passage index is used to obtain the list of passages, which are the input to the Filters. We adopt three different search engines:

- two engines based on the BM25 model [42], running on *keywords* and *lemmas* extracted from Wikipedia pages, respectively;
- one engine searching the DBpedia triples, as described in Section 5.1.

Filters: a pipeline of filters is adopted to assign a score to each text passage belonging to documents retrieved in the previous step:

- *Zero filter*: it removes passages having a score equal to 0;
- *Top-N filter*: it selects the N top-ranked passages, i.e. those with the highest score;

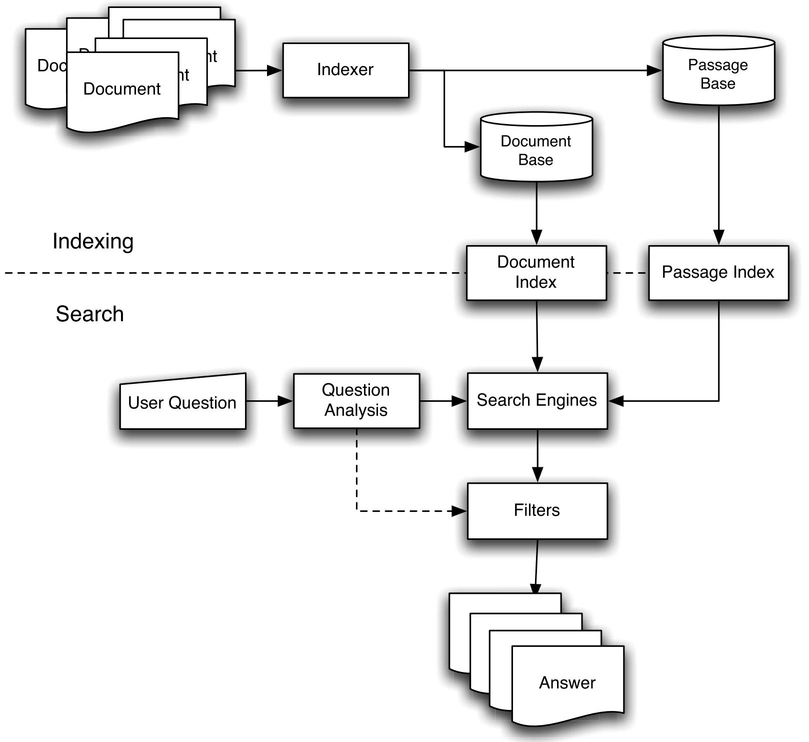


Fig. 3. QuestionCube architecture.

- *Terms filter*: it assigns a score to a passage based on the frequency of occurrence of question terms in the passage;
- *Exact sequence filter*: it assigns a score to a passage based on the number of terms that occur in the longest overlapping sequence between question and passage;
- *Normalization filter*: it assigns a normalized score to a passage based on the passage length. Both a simple normalization filter considering only the number of terms (called *Byte-size Normalization*) and a filter based on the *Pivoted Normalized Document Length* technique are implemented. More details in [33,50];
- *N-grams filter*: it assigns a score to a passage based on the overlapping of n-grams between the question and the passage;
- *Density filter*: it assigns a score to a passage based on the distance of the question terms inside that passage. The closer the question terms in the passage, the higher the score. The density is calculated using a modified version of the Minimal Span Weighting schema [38]:

$$\left(\frac{|\, q \cap d\, |}{1 + \max(mms) - \min(mms)}\right)$$

  where q and d are the set of terms in the query and in the document, respectively (specifically, the query is the question and the document is the passage); $\max(mms)$ and $\min(mms)$ are the initial and final location of the sequence of document terms containing all the query terms;
- *Distributional filter*: it assigns a score to a passage based on its similarity with the question, computed using a Distributional Semantic Model (DSM). DSM is built applying Latent Semantic Analysis to the term-term matrix of the Wikipedia pages. We construct a matrix containing the 100,000 most frequent terms in Wikipedia (stopwords are removed and multi-word expressions are not allowed). In DSMs, given the vector representation of two words $\mathbf{u} = (u_1, u_2, \ldots, u_n)^\top$ and $\mathbf{v} = (v_1, v_2, \ldots, v_n)^\top$ (we used vectors with 400 dimensions), it is always possible to compute their similarity as the cosine of the angle between them. However, the question and the passages are sentences composed by several terms, so in order to compute their similarity we need a method to compose words occurring in these sentences. It is possible to combine words through vector addition ($+$). This operator is similar to the superposition defined in connectionist systems [51], and corresponds to the point-wise sum of components. Addition is a commutative operator, which means that it does not take into account any order or underlying structures existing between words. More complex methods to combine word vectors might be exploited. Formally, if $q = q_1 q_2 \ldots q_n$ and $p = p_1 p_2 \ldots p_m$ are the question and the passage respectively, we build two vectors $\mathbf{q}$ and $\mathbf{p}$ which represent the question and the passage in a semantic

Leonardo da Vinci

Portrait of Leonardo by Melzi

Born	Leonardo di ser Piero da Vinci April 15, 1452 Vinci, Republic of Florence (present-day Italy)
Died	May 2, 1519 (aged 67) Amboise, Kingdom of France
Known for	Diverse fields of the arts and sciences
Notable work(s)	*Mona Lisa* *The Last Supper* *The Vitruvian Man* *Lady with an Ermine*
Style	High Renaissance
Signature	[illegible]

Fig. 4. Leonardo da Vinci infobox.

space respectively. Vector representations are built applying the addition operator to the vector representation of words belonging to them:

$$\mathbf{q} = q_1 + q_2 + \ldots + q_n$$
$$\mathbf{p} = p_1 + p_2 \ldots + p_m \tag{1}$$

The similarity between **q** and **p** is computed as the cosine similarity between them. This similarity is the score assigned to the passage;

- *Z-Score filter*: it assigns a score to a passage based on the Z-Score normalization [49] of scores assigned by the search engines and the other filters;
- *CombSum filter*: it assigns a score to a passage by summing the scores assigned by the search engines and the other filters [49].

Terms filter and *Density filter* have an enhanced version which adopts the combination of lemmas and part-of-speech tags as features, instead of terms. A boost factor can be assigned to each filter in order to increase (or decrease) its strength.

5.1. Using DBpedia as knowledge source

The question *When was Leonardo da Vinci born?* shows the difficulty to extract the correct answer, even though the information is contained in Wikipedia. In this case, the date of birth is "April 15, 1452", but it cannot be identified by adopting the classical passage retrieval process implemented by the Question Answering module. In order to manage this kind of questions, we used a specific search engine which leverages the knowledge contained in DBpedia. DBpedia includes structured information embedded in the Wikipedia articles – the "infobox". Fig. 4 reports the infobox for Leonardo da Vinci, which contains many useful information, such as birth date, death date, nationality, etc. DBpedia represents resources, properties as well as relations between resources using RDF triples, which contain three components: subject, predicate, and object. For example, the RDF triple that represents the birth date of Leonardo da Vinci is:

```
<http://dbpedia.org/resource/Leonardo\_da\_Vinci,
      dbpedia-owl:birthDate,
            1452-04-15.>
```

DBpedia allows to query relations and properties associated with Wikipedia resources, thus the birth date of Leonardo da Vinci can be found by accessing the property `dbpedia-owl:birthDate`. In order to leverage the knowledge contained in DBpedia, we have manually created a mapping between the 50 most frequent DBpedia properties and different lexicalizations, in question form, asking for those specific properties. For example, the property `dbpedia-owl:birthDate` is mapped to the questions *When was he born?*, or *What is the date of birth?*, and other similar wordings. In this way we have obtained two datasets containing 347 questions for Italian and 312 for English, where each question is tagged with the corresponding DBpedia property. Each dataset is used to train a Rocchio classifier [43] that, given a question, is able to predict the DBpedia property it is mapped to. The features used to train each classifier are all the words occurring in the questions. Stopwords are not removed, since words such as *When*, *How*, *Where* are useful hints to guide the classifier to the correct classification (performance details on the classifier are reported in Section 8.1.1).

In order to retrieve relevant information from DBpedia we queried an additional search engine containing documents which are the lexicalization of RDF triples with the same subject in the form: ⟨*label of the subject*, *label of the predicate*, *label of the object*⟩ (value of the object, in case of literals). The lexicalization for the previous example is: ⟨Leonardo da Vinci, date of birth, 1452-04-15⟩. Only triples related to the 50 selected properties are lexicalized in that way. Each document has an additional field reporting the DBpedia properties it contains, such as `dbpedia-owl:birthDate`. When a query (question) is sent to the DBpedia search engine, it is first classified using the Rocchio classifier in order to identify the property it refers to; then, the selected property is added to the query, along with the named entities (if any) occurring in the question. The query is submitted to the search engine, which retrieves the set of documents relevant to the query. Starting from the documents, the system extracts the corresponding list of passages (the RDF triples), which are scored using the pipeline of filters described in Section 5, and an additional DBpedia property filter. This filter scores the triples containing the property returned by the classifier with its confidence, and the triples not containing it with 0. It is worth to note that each question is always submitted to both search engines working on Wikipedia and DBpedia, in order to retrieve results from both knowledge sources.

6. Answer scoring

The main goal of the Answer Scoring module is to assign a score to each of the four possible answers to a question. Similarly to the approaches used in the context of the Answer Validation Exercise [45], we adopted five criteria based on the analysis of the passages returned by the Question Answering module. Each criterion returns a score for each possible answer, which is normalized using the sum of the scores of the four possible answers. More formally, given $\langle q, (A, B, C, D)\rangle$, each criterion computes $\langle q, (c_A, c_B, c_C, c_D)\rangle$, where c_X is the score assigned to the candidate answer X. In the following, each criterion is described by taking into account our running example.

Title Levenshtein (TL) criterion. It computes the Levenshtein distance between a candidate answer X and the title t_i of the Wikipedia page returned by the Question Answering module. The Levenshtein distance measures the difference between two strings, and is defined as the minimum number of single-character edits (i.e. insertion, deletion, substitution) required to change one string into the other. As the Levenshtein distance is a distance measure, rather than a similarity measure, we compute:

$$\frac{max(len(X), len(t_i)) - lev(X, t_i)}{max(len(X), len(t_i))}$$

where $len(\cdot)$ is the function computing the length of the string, and $lev(X, t_i)$ is the Levenshtein distance between X and t_i, used as a normalization factor (to have scores in the $[0, 1]$ interval). In our running example, taking into account just the first passage returned by the Question Answering module (Table 1), the answer B) Ridley Scott occurs in the title of the page containing the passage, so it gets the maximum score of 1, while the answer A) Harrison Ford gets a score equal to $\frac{13-12}{13} = 0.077$, the answer C) Philip Dick gets a score equal to $\frac{12-10}{12} = 0.167$, and the answer D) James Cameron gets a score equal to $\frac{13-12}{13} = 0.077$. The normalized score of each answer is:

$$c_A = \frac{0.077}{0.077+1+0.167+0.077} = 0.058$$
$$c_B = \frac{1}{0.077+1+0.167+0.077} = 0.757$$
$$c_C = \frac{0.167}{0.077+1+0.167+0.077} = 0.126$$
$$c_D = \frac{0.077}{0.077+1+0.167+0.077} = 0.058.$$

Longest Common Subsequence (LCS) criterion. It computes the Longest Common Subsequence between a candidate answer X and a passage of text p_i returned by the Question Answering module, or just the title t_i of the Wikipedia page the passage p_i is extracted from. In our running example, taking into account only the second passage, the answer A) Harrison Ford gets a score equal to 13, the answer B) Ridley Scott gets a score equal to 12, the answer C) Philip Dick gets a score equal to 11, and the answer D) James Cameron gets a score equal to 0 since it does not occur in the passage. The normalized score of each answer is:

$$c_A = \frac{13}{13+12+11+0} = 0.361$$
$$c_B = \frac{12}{13+12+11+0} = 0.333$$
$$c_C = \frac{11}{13+12+11+0} = 0.305$$
$$c_D = \frac{0}{13+12+11+0} = 0.$$

Overlap criterion. It computes the Jaccard index between the set of terms in a candidate answer X and the set of terms in a passage of text p_i returned by the Question Answering module. The Jaccard index measures the similarity between sets, and is defined as the size of the intersection divided by the size of the union of the sets. In our running example, taking into account only the second passage, answers A), B) and C) get a score of $\frac{2}{49} = 0.041$, while the answer D) gets a score equal to 0. The normalized score of each answer is:

$$c_A = c_B = c_C = \frac{0.041}{0.041+0.041+0.041+0} = 0.333$$
$$c_D = \frac{0}{0.041+0.041+0.041+0} = 0.$$

Exact Substring (ES) criterion. It computes the length in characters of the longest common substring between a candidate answer X and a passage of text p_i returned by the Question Answering module, normalized using the length of the candidate answer. In our running example, taking into account only the second passage returned by the Question Answering module, the answer A) Harrison Ford gets a score of $\frac{13}{13} = 1$, the answer B) Ridley Scott gets a score of $\frac{12}{12} = 1$, the answer C) Philip Dick gets a score of $\frac{6}{11} = 0.55$, and the answer D) James Cameron gets a score equal to 0. The normalized score of each answer is:

$$c_A = \frac{1}{1+1+0.55+0} = 0.392$$
$$c_B = \frac{1}{1+1+0.55+0} = 0.392$$
$$c_C = \frac{0.55}{1+1+0.55+0} = 0.215$$
$$c_D = \frac{0}{1+1+0.55+0} = 0.$$

Density criterion. It computes the density of the terms in a candidate answer X inside a passage of text p_i returned by the Question Answering module, using the modified version of the minimal overlapping span method [38] reported in Section 5. In our running example, taking into account only the second passage returned by the Question Answering module, the answers A) and B) get a score equal to 1, the answer C) gets a score equal to $\frac{2}{3} = 0.66$ (as the passage reports the full name Philip K. Dick, adding an extra token between the two tokens of the candidate answer), and the answer D) gets 0. The normalized score of each answer is:

$$c_A = \frac{1}{1+1+0.66+0} = 0.376$$
$$c_B = \frac{1}{1+1+0.66+0} = 0.376$$
$$c_C = \frac{0.66}{1+1+0.66+0} = 0.248$$
$$c_D = \frac{0}{1+1+0.66+0} = 0.$$

Each criterion has some parameters that can be set:

1. the *number of processed passages*: in this case the score of each answer is computed for each of the top-n passages returned by the Question Answering module, and the final score is the average of those values;
2. the *use of the weight w_i of the passages* returned by the Question Answering module: in this case the average computed at the previous point is weighted using the score w_i of each passage. This strategy allows to assign higher weights to passages deemed as more relevant to the question by the Question Answering module;
3. the *level of linguistic analysis* to adopt for processing the passages returned by the Question Answering module. Passages are represented by using keywords, lemmas, or stems, with or without stopword removal;
4. the *use of the question expansion*: the system asks four different questions obtained by concatenating the original one with each of the four possible candidate answers. In our running example, the virtual player queries the Question Answering module using the following questions *Who directed Blade Runner? Harrison Ford*, *Who directed Blade Runner? Ridley Scott*, *Who directed Blade Runner? Philip Dick* and *Who directed Blade Runner? James Cameron*. If a criterion adopts the question expansion, it uses four different sets of passages (one for each question), instead of using the same set of passages.

Besides the above mentioned criteria for Answer Scoring, the distributional filter (Section 5) could be also adopted for comparing answers against the retrieved passages. We finally decided to avoid using that filter because, after the selection of the most relevant passages to a specific question, we expect that the candidate answer is already contained in one of the passages, and the filters based on pure lexical comparison are sufficient to properly select and score it. Moreover, the answers usually contain one or two words, and most of them are named entities, numbers, or dates, which could be hardly identified by a distributional filter.

It is worth to mention how we managed a specific type of questions, i.e. those formulated in negative form, such as *Quale di questi super-criminali non e' uno storico nemico di Batman?* (in English *Which of these villains is not a historical enemy of Batman?*). Let us suppose that the candidate answers are *Dottor Destino*, *Mister Freeze*, *Pinguino*, and *Joker*. The Question Answering module returned passages containing *Joker*, *Pinguino*, and *Mister Freeze*, that would be correct if the question was raised in positive form. In fact, when the question is raised in negative form, the top-ranked passages often contain the candidate answers that are actually those to be discarded. For this reason, when the system detects a negative question, the strategy for scoring the candidate answers is reversed, meaning that those with the highest score become the less plausible and vice versa. The process of detecting questions raised in negative form leverages several heuristics based on the use of regular expressions. Performance details on the reverse scoring method and the regular expressions at correctly detecting negative questions are reported in Section 8.1.1.

7. Decision making

The Decision Making module is responsible for the decision about answering to a specific question, retiring from the game or using one of the available lifelines. The decision strategy evaluates the uncertainty of the information provided by the Question Answering module and the Answer Scoring, the available lifelines, and the level of the question in order to take the final decision. The decision making strategy we devised encapsulates two heuristics to manage the following situations of uncertainty:

1. the maximum score computed by the Answer Scoring for the four candidate answers is very low. This means that either the quality of the passages retrieved by the Question Answering module may be low, hence those passages are not useful to find the correct answer to the question, or the criteria adopted by the Answer Scoring for assigning a score to the candidate answers are not satisfactory;
2. the difference between the score of the best candidate answer and the second best candidate answer is very small. This means that the virtual player is not able to provide clear evidence of the most likely candidate answer between the two candidate answers.

In a situation in which the virtual player has enough *confidence* in one of the candidate answers, it can answer the question without using any lifeline. In a situation of uncertainty, the decision making algorithm can take one of the following decisions: 1) to retire from the game; 2) to use one or more available lifelines; 3) to continue to play by providing a random answer. The random strategy can be also useful in some situations, in particular at specific levels of the game (6th or 11th question) when a wrong answer causes no loss in the earned amount.

The decision making algorithm implements a simple strategy for managing the lifelines, meaning that the order of their usage is independent of the level of the current question. The system leverages first the *Poll the Audience* lifeline and, if it does not return a candidate answer with a good confidence, the *Phone a Friend* is explored, and finally *50:50*. Other authors faced the problem of defining a more dynamic decision making strategy for the WWBM game. In [25], the decision making module constructs a decision tree that encodes the probabilities and utilities at each potential future state of the game. The tree consists of both decision forks for choosing whether to answer the question, to use a lifeline, or to walk away, and chance forks to encode the uncertainty of correctly answering the questions. The best choice is obtained with the action that maximizes the expected utility. The strategy is able to parameterize the risk of the virtual player, which can exhibit a risk averse behavior or a more risk neutral one. Probabilities are assigned to the nodes of the tree based on historical past performance on a sample of questions from the associated difficulty level. A different strategy based on dynamic programming is adopted in [41] to analyze two different objectives: 1) to maximize the expected reward, 2) to maximize the probability of reaching a given question. An analysis of the results presented in that work allowed us to define the order in which our decision making module should check and use the available lifelines. More specifically, we found out that *Poll the Audience* must be checked before the others, while no specific indications are provided for *50:50* and *Phone a Friend*.

The functions used by our decision making algorithm (Algorithm 1) are the following:

- Best(Answers) and SecondBest(Answers) return the best and second best candidate answer to a question q, respectively;
- CanUse(lifeline) returns *true* if that specific *lifeline* has not yet been used in the current game, otherwise it returns *false*;
- Use(lifeline) returns a new set of answers together with their scores obtained by adopting that specific *lifeline*;
- CanRisk() returns *true* if the current question allows the player to provide a wrong answer without losing the earned money (6th or 11th question), otherwise it returns *false*;

Algorithm 1 Decision making algorithm.

```
1: procedure DECISION MAKING($< q, (c_A, c_B, c_C, c_D) >$, lifelines)  ▷ Decision strategy based on the scores of the four candidate answers for question $q$, and the available lifelines
2:     BestAnswer ← BEST($< q, (c_A, c_B, c_C, c_D) >$)
3:     SecondBestAnswer ← SECONDBEST($< q, (c_A, c_B, c_C, c_D) >$)    ▷ Selection of the best and second best candidate answers
4:     if BestAnswer.score = 0
   or (BestAnswer.score − SecondBestAnswer.score)
   < (BestAnswer.score ∗ threshold) then
          ▷ Situation of uncertainty: system not confident in the best answer, or undecided between the best and second best candidate answer
5:         if CANUSE(Poll the Audience) then
6:             audienceAnswers ← USE(Poll the Audience)
7:             lifelines ← lifelines − {Poll the Audience}
8:             if audienceAnswers.score > 0 then
9:                 RETURN BEST(audienceAnswers)
10:            end if
11:        end if
12:        if CANUSE(Phone a Friend) then
13:            friendAnswer ← USE(Phone a Friend)
14:            lifelines ← lifelines − {Phone a Friend}
15:            if friendAnswer ≠ null then
16:                RETURN friendAnswer
17:            end if
18:        end if
19:        if (CANUSE(50:50) and CANRISK()) then
20:            50:50answers ← USE(50:50)
21:            lifelines ← lifelines − {50:50}
22:            if 50:50answers.score > 0 then
23:                RETURN BEST(50:50answers)
24:            else
25:                RETURN RANDOM(50:50answers)
26:            end if
27:        end if
28:        if CANRISK() then
29:            RETURN RANDOM(answers)    ▷ No more lifelines but the player can risk
30:        end if
31:        RETIRE()
32:    else
33:        RETURN BestAnswer
34:    end if
35: end procedure
```

- RANDOM(ANSWERS) allows the virtual player to provide a random answer to a question;
- RETIRE() allows the virtual player to retire and win the earned money.

The virtual player uses no lifelines if it has enough confidence in one of the four answers (`if` statement at step 4). In this case the provided answer is the one with the highest score (step 33). If the virtual player is in a situation of uncertainty, it explores the available lifelines starting by *Poll the Audience* (steps 5–11). If the player has enough confidence in the answer provided by that lifeline (step 8), it returns that answer (step 9), otherwise it continues to explore the other available lifelines. Steps 12–18 manage the *Phone a Friend* lifeline, which allows to return the answer possibly provided by a friend (step 16). The usage of the *50:50* lifeline is related to the level of the game reached by the player. If the user can risk, meaning that the money would not be lost even providing a wrong answer to the question (step 19), the player uses the *50:50* lifeline (step 20). The player chooses the answer with the highest score (step 23) if it has enough confidence in the answer, otherwise it randomly returns one of the two remaining answers (step 25). If the player cannot rely on lifelines, or it does not have enough confidence in the answers provided by lifelines, then it returns a randomly chosen answer in case the user can risk (steps 28–30), otherwise it retires from the game (step 31).

Given that the game is played using a board game variant, we implemented a specific strategy to simulate the actual way of using lifelines, as their use in the board game implies the interaction with the other players. *50:50* is simulated in the same way as in the real game, i.e. by removing two wrong answers among the four candidate answers. As in the real game, *Phone a Friend* and *Poll the Audience* lifelines are not always able to return the correct answer. As regards *Phone a Friend*, it is worth to notice that usually the higher the level of the game, the more difficult is to answer to the question. Hence, this lifeline works as follows: it always returns the correct answer when used for levels from 1 to 5; it randomly chooses between two alternatives, i.e. providing the correct answer or not returning the answer at all, when used for levels from 6 to 10; it randomly chooses between three alternatives, i.e. providing the correct answer, not returning the answer at all or returning the wrong answer, when used for levels from 11 to 15. On the other side, *Poll the Audience* is simulated in order to distribute the votes coming from the audience (in percentage) among the candidate answers. Without a real audience, we simulated the distribution of votes using a strategy which takes into account both the current level of the question and a degree of randomness. We first assign a percentage of all votes to the correct answer, representing the percentage of the

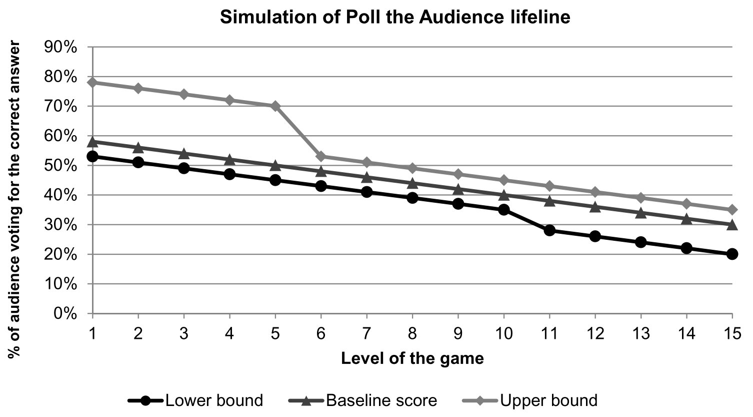


Fig. 5. Percentage of the audience voting for the correct answer when *Poll the Audience* lifeline is used.

audience that would know the correct answer, then we randomly distribute the remaining votes among the other candidate answers. The percentage of votes assigned to the correct answer – baseline – is inversely proportional to level of the game, i.e. the more difficult the question, the lower the confidence; then, the baseline percentage of votes is randomly perturbed according to the level of the game, between the lower and upper bound depicted in Fig. 5. It is worth noting that the perturbation of the baseline percentage is different for questions whose level is between 1 and 5, 6 and 10, 11 and 15. For example, if the player uses *Poll the Audience* at level 3, the baseline percentage is equal to 54%, and it is randomly perturbed between 49% and 74%; the remaining votes (between 51% and 26%) are randomly distributed among the remaining answers; if the lifeline is used at level 14, the baseline percentage is 32% and it is randomly perturbed between 22% and 37%, and the remaining votes (between 78% and 63%) are randomly distributed among the remaining answers.

8. Experimental evaluation

The goal of the evaluation is twofold:

1. to assess the effectiveness of the Question Answering and Answer Scoring modules to answer to questions of the game, and compare the results with those obtained by human players. The experiment is discussed in Section 8.1;
2. to evaluate the accuracy of the virtual player to play the game, taking also into account the strategy implemented by the Decision Making module, and to compare the results with those obtained by human players. The experiment is discussed in Section 8.2.

We used two datasets: one containing questions from the WWBM official Italian boardgame, and one containing questions from the English boardgame.[3] Both datasets contain 1960 questions, subdivided in 15 groups, one for each level of the game. The first 10 levels contain 160 questions each, while levels 11, 12, 13, 14 and 15 contain 120, 90, 70, 50 and 30 questions, respectively. 113 questions for Italian, and 84 for English are in negative form. The distribution of negative questions per level of the game is reported in Fig. 6.

While previous results to compare with are available for the English version of the game (even though not on the same dataset), we are not aware of previous results for Italian, i.e. we do not have any baseline to compare with. The metric adopted for the evaluation is the *accuracy*, i.e. the proportion of correctly answered questions, computed as the ratio between the number of correct answers (n_c) and the total number of questions (n): $accuracy = \frac{n_c}{n}$. The significance of the results is assessed using McNemar's test with the application of the Bonferroni correction.

8.1. Experiment 1: evaluation of the performance of QA and answer scoring

The goal of this experiment is to assess the accuracy of the five criteria for answer scoring, properly configured using the parameters described in Section 6, namely:

- number of processed passages: we tested 10 configurations, by using the top-n, $n = 1, 2, 3, 4, 5, 10, 15, 20, 25, 30$ passages returned by the QA module;
- use of the score w_i for the passages returned by the QA module: we tested 2 configurations, *using* and *not using* the weight for each passage returned by the QA module;

[3] Datasets available by contacting one of the authors.

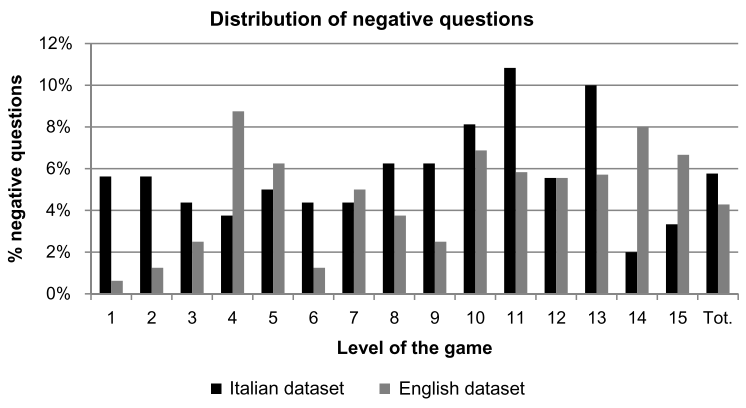


Fig. 6. Distribution of negative questions per level of the game.

Table 2
Performance of the top-15 configurations (averaged over all the questions). Acronyms: **P** # of passages, **Lex** Lexicalization, **KWD** Keywords, **LEM** Lemmas, **ST** Stems, **Score** Use of the score of the passage, **SW** Use of stopword removal, **QE** Use of the question expansion.

Rank	Criterion	P	Lex	Score	SW	QE	Accuracy
Italian dataset							
1	Overlap	25	ST	Y	Y	N	64.29%
2	Overlap	25	LEM	Y	Y	N	64.29%
3	Density	3	KWD	Y	N	Y	64.03%
4	Density	30	ST	Y	Y	N	64.03%
5	Density	30	LEM	Y	Y	N	64.03%
6	Overlap	20	ST	Y	Y	N	63.78%
7	Overlap	20	LEM	Y	Y	N	63.78%
8	Overlap	30	ST	Y	Y	N	63.78%
9	Overlap	30	LEM	Y	Y	N	63.78%
10	Density	20	ST	Y	Y	N	63.27%
11	Density	20	LEM	Y	Y	N	63.27%
12	Density	25	KWD	Y	Y	N	63.01%
13	Overlap	15	ST	Y	Y	N	62.76%
14	Overlap	15	LEM	Y	Y	N	62.76%
15	Overlap	20	ST	N	Y	N	62.76%
English dataset							
1	Overlap	25	LEM	Y	Y	N	59.47%
2	Overlap	25	ST	Y	Y	N	59.38%
3	Density	3	KWD	Y	N	Y	59.26%
4	Overlap	20	ST	Y	Y	N	59.22%
5	Density	30	ST	Y	Y	N	59.08%
6	Density	30	LEM	Y	Y	N	59.08%
7	Overlap	30	ST	Y	Y	N	58.99%
8	Density	20	ST	Y	Y	N	58.84%
9	Overlap	15	LEM	Y	Y	N	58.72%
10	Density	25	KWD	Y	Y	N	58.37%
11	Overlap	30	LEM	Y	Y	N	58.35%
12	Density	20	LEM	Y	Y	N	58.21%
13	Overlap	20	LEM	Y	Y	N	58.14%
14	Overlap	20	ST	N	Y	N	57.99%
15	Overlap	15	ST	Y	Y	N	57.97%

- level of linguistic analysis adopted to process the passages returned by the QA module: we tested 3 configurations, which represent passages using *keywords*, *stems*, or *lemmas*. We also run 2 further configurations, *using* and *not using stopwords removal*.
- use of question expansion: we tested 2 configurations, *with* and *without question expansion*.

Overall, we have a set of 1200 different configurations used to assign a score to each of the four possible answers to a question of the game: starting from these scores, the answer with the highest one is selected. The Question Answering system was configured using all the filters listed in Section 5.

Table 2 reports the accuracy of the best 15 configurations, averaged over the whole set of 1960 questions of the Italian and English datasets.

Table 3
Best and worst performance of each single criterion along with its configuration (averaged over all the questions). Acronyms: **P** # of passages, **Lex** Lexicalization, **Score** Use of the score of the passage, **SW** Use of stopword removal, **QE** Use of the query expansion, **ES** Exact Substring, **LCS** Longest Common Subsequence, **TL** Title Levenshtein, **KWD** Keywords, **LEM** Lemmas, **ST** Stems.

Rank	Criterion	P	Lex	Score	SW	QE	Accuracy
Italian dataset							
BEST	Overlap	25	ST	Y	Y	N	64.29%
WORST	Overlap	1	KWD	Y	N	N	42.60%
BEST	Density	3	KWD	Y	N	Y	64.03%
WORST	Density	1	LEM	N	N	N	43.37%
BEST	ES	30	KWD	Y	Y	N	59.18%
WORST	ES	1	KWD	Y	N	N	42.09%
BEST	LCS	3	KWD	Y	N	Y	57.14%
WORST	LCS	1	LEM	Y	Y	N	41.07%
BEST	TL	1	KWD	Y	Y	Y	40.05%
WORST	TL	1	LEM	Y	N	N	20.15%
English dataset							
BEST	Overlap	25	LEM	Y	Y	N	59.47%
WORST	Overlap	1	KWD	Y	N	N	37.74%
BEST	Density	3	KWD	Y	N	Y	59.26%
WORST	Density	1	ST	Y	N	N	38.58%
BEST	ES	30	KWD	Y	Y	N	54.62%
WORST	ES	1	KWD	Y	N	N	37.46%
BEST	LCS	3	KWD	Y	N	Y	52.19%
WORST	LCS	1	LEM	Y	Y	N	36.04%
BEST	TL	1	KWD	Y	Y	Y	35.17%
WORST	TL	1	LEM	N	N	N	15.12%

It is worth to note that *Overlap* and *Density* are the best performing criteria for both the Italian and the English datasets (the top-85 best configurations are obtained using those criteria). The analysis of the results unveils the usefulness of taking into account a considerable number of passages, the usefulness of leveraging the score of the passages returned by the QA module, and the effectiveness of the process of stopwords removal. Finally, the question expansion process seems to negatively affect the accuracy of the system. The worst criterion is *Title Levenshtein* for both Italian and English: indeed, the worst 100 results are obtained by that criterion.

In order to have a clear picture of the accuracy of the whole set of configurations, Table 3 reports, for each criterion, its best and worst configuration.

We observe that the best and worst configurations of each criterion for Italian and English are pretty much the same, even though the accuracy obtained on the English dataset is 5% lower than that obtained for Italian, on average. Statistical tests show that there is no significant difference between the best configuration of *Overlap* and the best configuration of *Density*, but their accuracy is significantly better than the best configurations of the other criteria, regardless of the language ($p < 0.01$).

Starting from the accuracy of the single configurations, we tried to combine them in order to improve the overall accuracy of the Answer Scoring. Differently from the greedy combination adopted in [37], in this work we propose a combination based on a pointwise learning to rank approach using regression-based algorithms [29], in which question-answer pairs (q, a) are labeled with the relevance judgments of the answer a with respect to the question q. In our setting, the correct answer to a question of the game is labeled with the relevance judgment 1, while the other three incorrect answers are labeled with 0. Each training example is represented using a feature vector consisting of the *level of the question (from 1 to 15)*, and all the *1200 single scores* obtained by the above mentioned configurations. We opted for using Random Forests (RF) [7] algorithm, an ensemble learning method, combining different *tree regressors* built using different samples of the training data and random subsets of data features. The final result is obtained by averaging the output of the single trees. The use of different data samples from the same distribution and of different feature sets for learning the individual trees prevent overfitting. We adopted the implementation provided by the RankLib library.[4]

Questions in each dataset are split into a training set *Tr*, and a test set *Ts*. The methodology adopted for obtaining *Tr* and *Ts* was the stratified 5-fold cross validation, where the stratification process ensures each fold contains the same distribution of questions for the different levels of the game. Given the size of each dataset (1960 questions), applying 5-fold cross validation means that the dataset is divided into 5 disjoint partitions, each containing 392 questions. The experiment was performed in 5 steps. At each step, 4 partitions were used as training set *Tr* (1568 questions), whereas the remaining partition was used as test set *Ts*. The steps were repeated until each of the 5 disjoint partitions was used as the *Ts*, and

[4] http://sourceforge.net/p/lemur/wiki/RankLib/.

Table 4
Error analysis for Italian and English.

Error type	%
Questions whose concepts do not occur in Wikipedia/DBpedia	13%
Questions whose concepts are not explicit in Wikipedia/DBpedia	15%
Questions which involve numbers, time and math	13%
Questions which require language proficiency	14%
Questions which require to make a comparison	7.50%
Questions which require knowledge about visual properties	7.50%
Other errors	30%

results were averaged over the 5 runs. In order to tune the parameters of the learning to rank (number of trees, learning rate, subsampling rate), we also used a *validation* set, obtained by sampling questions from the training set of each run. We sampled 12.5% of the training set (196 questions), and even in this case we took into account the distribution of the questions among the different levels (stratification). To sum up, at each step, the training set contains 1372 questions, the validation set contains 196 questions, and the remaining 392 questions are used as test set. The final accuracy obtained by combining *all* the individual configurations through the learning to rank strategy is equal to 79.64% for Italian, and 76.41% for English (averaged over the five runs), which is significantly better ($p \ll 0.0001$) than the accuracy of the best single configuration (64.29% for Italian, 59.47% for English). The result obtained for English is similar to that achieved in [25], in which the system correctly answered about 75% of questions, even though a different and smaller dataset was used. More details about the accuracy for each different level of difficulty of the game is presented and discussed in Section 8.1.3.

8.1.1. *Unanswerable questions and error analysis*

In order to understand the questions for which the system is not able to provide a correct answer, we classified them in specific categories, and we performed a proper error analysis. The following list contains categories of questions which remain unanswerable by our system:

- *Questions regarding concepts not occurring in Wikipedia/DBpedia:* some of these questions concern astrology, proverbs and sayings, and religion.
- *Questions regarding concepts which are not explicit in Wikipedia/DBpedia:* the system may not be able to provide an answer to some questions, even though the necessary information is contained in the knowledge sources. For example, the question *Quale di questi attori non e' figlio d'arte?* (in English *Which of these actors is not an actor son of an actor father?*) would require to match the concept *figlio d'arte* (actor son of an actor father), which is not explicit in Wikipedia/DBpedia.
- *Questions whose answers involve special numbers, periods of time, and mathematical computations:* the system fails to provide an answer when it contains Roman numerals, or it implies the computation of time frames, or some mathematical computations. For example, the candidate answers to the question *In quale secolo fu costruita la prima penna a sfera?* (in English *In what century was the first ballpoint pen built?*) are A) XX B) XVII C) XVIII D) XIX, but the system is not able to find a match with any of the candidate answers; the question *How long was William Harrison in office as the ninth president of USA?* involves the computation of a time frame which the system is not able to carry out; the question *How many degrees are each of the other two angles in an isosceles triangle, if one angle is* 120°*?* requires that the system knows that the total of all angles in any Euclidean triangle would sum to 180°.
- *Questions that require language proficiency:* questions such as *Nella lingua latina, il vocativo plurale e' sempre uguale a che cosa?* (in English *In Latin language, the vocative plural is always equal to what?*), would require a specific knowledge which is not encoded in Wikipedia/DBpedia.
- *Questions that require to make a comparison:* questions such as *Quale tra queste regioni italiane ha la superficie minore?* (in English *Which of these regions of Italy has the lowest surface?*) would require to make a comparison and the selection of the minimum value.
- *Questions that require knowledge of visual properties:* questions such as *Quale di queste squadre di calcio non ha come simbolo un esemplare di lupo?* (*Which of these football teams does not have a wolf as symbol?*) would require the knowledge of visual properties.

The error analysis carried out on the results of the experiments in the previous section is depicted in Table 4. 30% of the other errors are due to the wrong scoring of answers or to other factors, such as the heuristics to recognize and manage negative questions (Section 6). Hence, in order to evaluate how much effective that strategy is, we have evaluated on one hand the accuracy of regular expressions at correctly detecting negative questions, and on the other hand the effectiveness of the reverse scoring method. The accuracy of the heuristics based on the use of regular expressions is $F1 = 89.23\%$ for the Italian dataset, and $F1 = 98.20\%$ for the English one, while the reverse scoring method was evaluated by taking into account the percentage of negative questions for which the reverse scoring strategy was useful (i.e. the answer with the lowest score was correct), harmful (i.e. the answer with the lowest score was wrong) or indifferent (i.e. the correct answer is neither the one with the highest score nor the one with the lowest score): for the Italian dataset percentages are 80%,

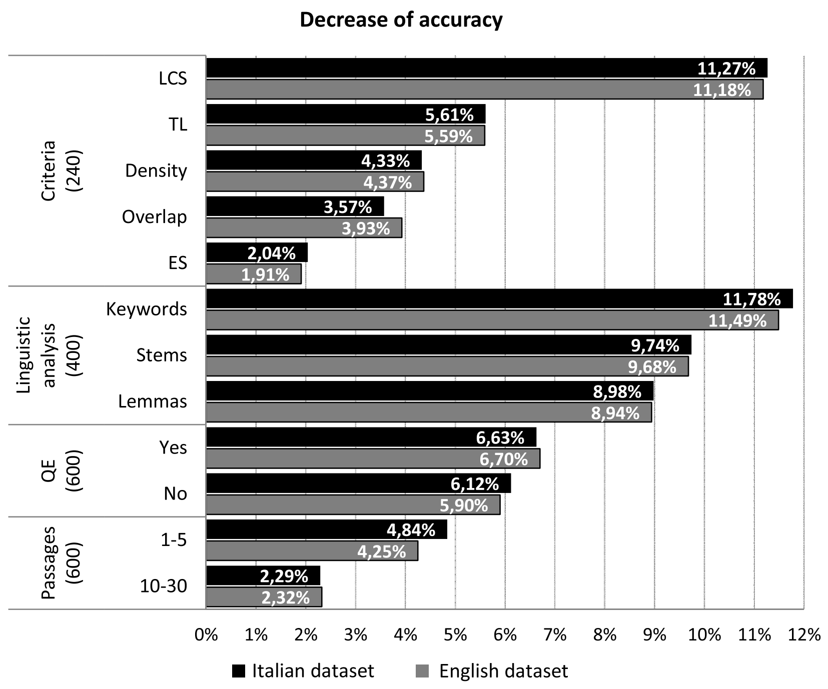


Fig. 7. Decrease of accuracy for ablation of feature groups. For each group, the number of features removed is also reported.

9%, and 11%, respectively, while for the English dataset percentages are 96%, 1%, and 3%, respectively. This means that the reverse scoring method is effective in practice.

Finally, in order to have some details about the DBpedia benefit, we have computed the number of questions for which the QA module returned passages coming from DBpedia. 290 questions for the Italian dataset, and 293 for the English one rely on passages coming from DBpedia (besides those coming from Wikipedia), and this means that the answer for those questions could potentially be extracted from DBpedia. For the sake of completeness we have also evaluated the accuracy of the Rocchio classifier described in Section 5.1, which allows to map different lexicalizations of questions asking for a specific property to the corresponding DBpedia property. The accuracy of the classifier, computed using leave-one-out is 85.59% for the English dataset, and 83.57% for the Italian one.

8.1.2. Ablation tests

To gain insights about the power of the different parameters used to configure each answer scoring criterion, we performed feature selection using *ablation tests*, by removing single features or groups of features.

As regards the ablation of single features, we trained a different model by removing each feature related to each single configuration, and measuring the corresponding predictive accuracy of the learning to rank model (i.e. the process is repeated 1200 times). The analysis of results unveils that only 160 out of 1200 times the accuracy of the learned models is lower than that obtained by the best configuration, regardless the language adopted. The maximum decrease of accuracy is 4.84% for Italian, and 5.00% for English. It is interesting to note that in each one of those 160 learned models a feature corresponding to a configuration adopting keywords for representing passages is removed. Hence the signal brought by *keywords* is more relevant than those brought by stems and lemmas.

We also performed ablation tests by removing *groups* of features. We defined the following twelve different groups:

- Five groups, each corresponding to a different answer scoring criterion, i.e. each group contains all the features corresponding to all the configurations of each single criterion;
- Three groups, each corresponding to a different level of linguistic analysis adopted for processing the passages of text returned by the QA module;
- Two groups, each containing all the configurations using and not using the question expansion strategy;
- Two groups, each containing all the configurations using a number of passages from 1 to 5, and from 10 to 30, respectively.

Fig. 7 reports the decrease of accuracy obtained by different models trained by removing each single group of features. Groups with higher values are better, meaning that those features have a higher impact on the overall performance of the model, since their ablation leads to a higher decrease of accuracy. All the results are statistically significant when compared to the best configuration obtained by the learning to rank strategy ($p \ll 0.0001$).

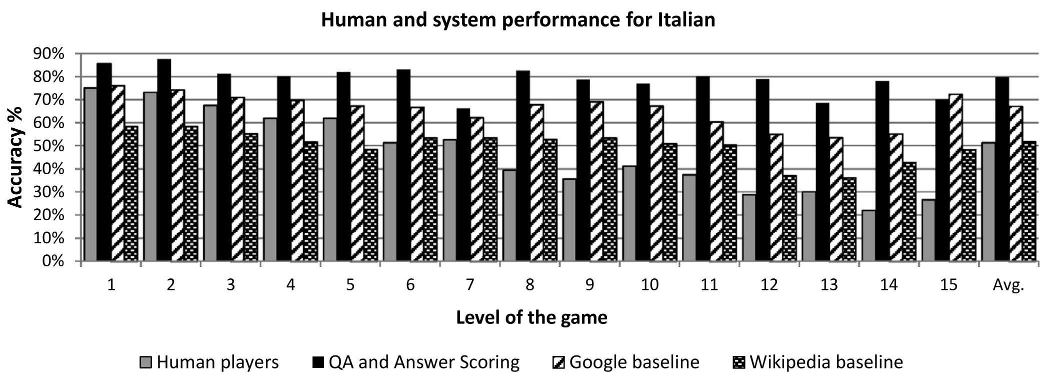


Fig. 8. Per level accuracy of system and human performance for Italian.

As regards the groups corresponding to the answer scoring criteria, the best performance is obtained by LCS, followed by TL, Density, Overlap and ES. A comparison with the performance of the single configurations reported in Table 2 and Table 3, shows that *Overlap* and *Density* criteria are the best performing individually, but when they are individually removed, the overall performance is not significantly different. This supports the finding that the signal they bring is very overlapping. On the other side, the TL criterion was the worst performing individually but, when individually removed, it leads to a decrease of the overall accuracy greater than 5%. As regards the linguistic analysis adopted for processing the passages of text returned by the QA module, keyword-based representations have the best performance, followed by stems and lemmas which behave in a similar way. This was already observed by removing one configuration at a time, where the ablation of configurations using representations based on stems or lemmas did not lead to a decrease of accuracy. As regards the question expansion mechanism, methods adopting it show a slightly better performance (less than 1%) than those not adopting it, and this is in line with the performance of the single configurations, in which we observed that the question expansion does not seem to have impact on the accuracy of the system. Finally, the configurations using a fewer number of passages (from 1 to 5) are better than those using more passages (from 10 to 30). This seems to contradict the finding stemmed from the analysis of the top-15 configurations (Table 2), where the best performance were obtained using 15 to 30 passages. A possible interpretation is that the information overlap existing using 15 to 30 passages is higher than that existing using 1 to 5 passages, and this led the former combination to be less effective than the latter.

8.1.3. Per level analysis of system and human performance

We compared the performance of QA and Answer Scoring with that of human players to provide answers to questions at different levels of difficulty.

Playing successfully the WWBM game heavily depends on the player's knowledge about popular culture, hence the comparison with the human players is only performed for the Italian dataset. To this purpose we involved 98 human players, selected using the availability sampling strategy [46] (Italian students or graduates). The dataset of 1960 Italian questions was randomly split into 98 disjoint sets of 20 questions each; each set was assigned to a different user, who provided the answers without having the possibility to consult the Web or other knowledge sources (the level of each question was not disclosed to the users).

As baseline we queried Google with each question and we retrieved the top-30 snippets of text returned by the search engine. Hence, we computed a score for each candidate answer by multiplying the number of times the answer occurred in each snippet with the inverse of the rank of the snippet. Finally, we selected the candidate answer with the highest score (a random selection was adopted to break ties). We refer to this as *Google baseline*.

As Google queries the whole Web, while we use only information available in Wikipedia, we decided to compare against a fairer baseline as well. We queried Google by restricting the results only to pages coming from Wikipedia, retrieving the top-30 snippets and scoring them in the same way as for the *Google baseline*. We refer to this as *Wikipedia baseline*.

Figs. 8 and 9 report, for each level of the game:

1. the accuracy of the best configuration obtained by the learning to rank strategy for Italian and English, whose average accuracy is 79.64% ($\sigma = 6.07\%$) for Italian and 76.41% ($\sigma = 1.45\%$) for English;
2. the accuracy achieved by the human players, which is 51.33% ($\sigma = 17.61\%$) for Italian;
3. the accuracy of the Google baseline, which is 67.13% for Italian and 71.80% for English;
4. the accuracy of the Wikipedia baseline, which is 51.71% for Italian and 60.65% for English.

The accuracy of the system is significantly better than the *Google baseline*, the *Wikipedia baseline* and the accuracy of human players ($p \ll 0.0001$).

The system has quite similar performance for all the levels of the game. As regards the Italian dataset, the best performance is obtained for level 2, while the worst is obtained for levels 7 and 13. However, the error analysis did not report

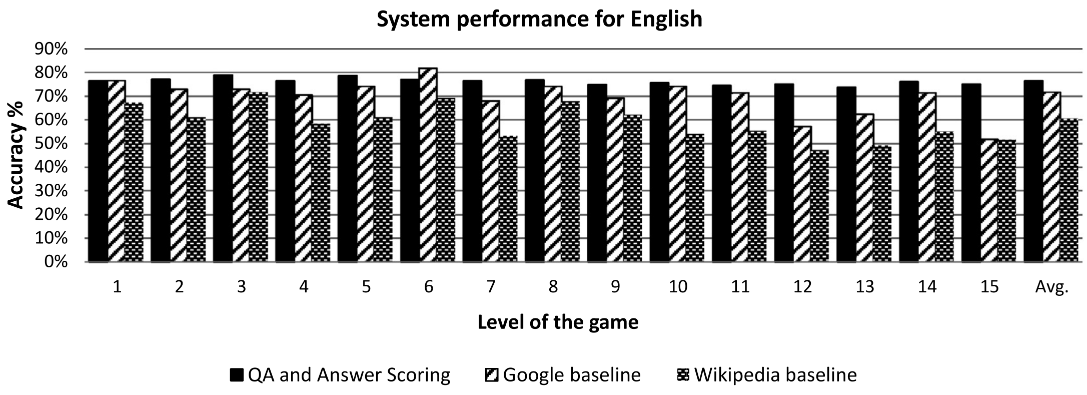


Fig. 9. Per level accuracy of system performance for English.

any specific problem for those levels: we only observed a higher concentration of questions the system is not able to deal with (see Section 8.1.1) in one of the five folds. As regards English, the performance is almost constant for all the levels of the game (indeed the standard deviation is very low).

Accuracy of human players (for Italian) decreases almost monotonically, with the best performance obtained on the first level of the game, and the worst on level 14. This observation is coherent with the fact that lower levels of the game correspond to easier questions, while higher levels correspond to more complex questions for which a deeper knowledge is required to provide the correct answers.

The system (for Italian) significantly outperforms humans for all the 15 levels of the game (not only on average). It is worth to analyze the performance of the players for groups of questions from 1 to 5, 6 to 10, and 11 to 15, respectively, when they reach the guarantee points where the money earned is banked. As regards the first group of questions, humans obtain the worst performance on the (fourth and) fifth question (61.88%), and this is not surprising since this is the first point in which players have the guarantee to keep the earned money. Even though the level of the questions is not disclosed to human players, our hypothesis is that milestone questions are likely more difficult, due to the way the game is designed. Surprisingly, this behavior is not observed for the second milestone question, albeit the performance on that question is below the average. Results obtained by humans on the last group of questions are all below the average, and not comparable with those obtained by the system (absolute difference in performance ranging from +38% to +56%).

The Google baseline is always better than human players (for Italian), but it is less accurate than the system (absolute difference in performance amounts to −12.51% for Italian, and −4.61% for English, on average). The Wikipedia baseline is less accurate than the Google baseline for both Italian and English, and is just slightly more accurate than human players on average (for Italian).

To sum up, the virtual player built using the QA and Answer Scoring modules has the potential to beat human players when playing a real game with its rules, since it is able to correctly answer to questions at different levels of difficulty. We observe very similar performance regardless the language, i.e. the Question Answering framework and the Answer Scoring criteria work in the same way for Italian and English. The small differences in performance are likely due to the different number of documents extracted from Wikipedia (almost one million for Italian, and three millions for English), which could have impact on the performance of the search engines in the QA framework, and of course to the fact that the two datasets are not comparable.

However, playing a real game is a very complex task, since it requires a proper strategy to manage the lifelines, to decide whether to answer to a question or to retire from the game by taking the earned money. This is the purpose of the experiment described in the next section.

8.2. Experiment 2: evaluation of the virtual player

The goal of this experiment is to evaluate the ability of the virtual player to play the game, by implementing a proper strategy to use the lifelines, to decide whether to answer to a question even in a condition of uncertainty or to retire from the game by taking the earning money. We ran the experiment by comparing the performance of the human players with the performance achieved by the best system configuration obtained by using the learning to rank approach, and whose strategy to play the game is defined by the decision making algorithm described in Section 7. The comparison is carried out by evaluating the level reached during the game and the money earned by the players.[5] As in the previous experiment, the comparison between the performance of the virtual player and humans is carried out only for Italian.

[5] The Italian and English versions of the game have a different currency and a different distribution of monetary values for each level of the game. For the sake of comparison we decided to use the same currency and monetary values as for Italian.

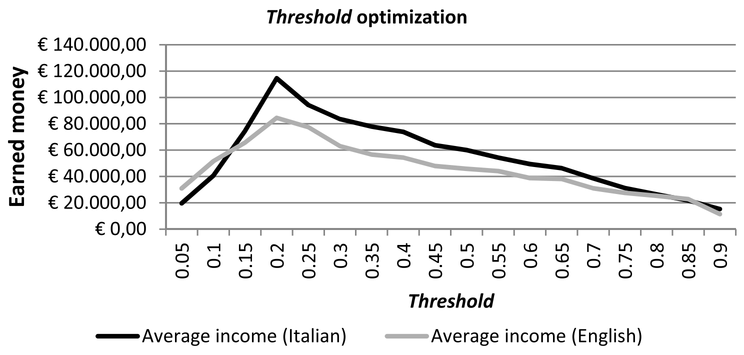


Fig. 10. Plot of the average income for different values of the threshold measuring the difference between the score of the best and the second best candidate answer.

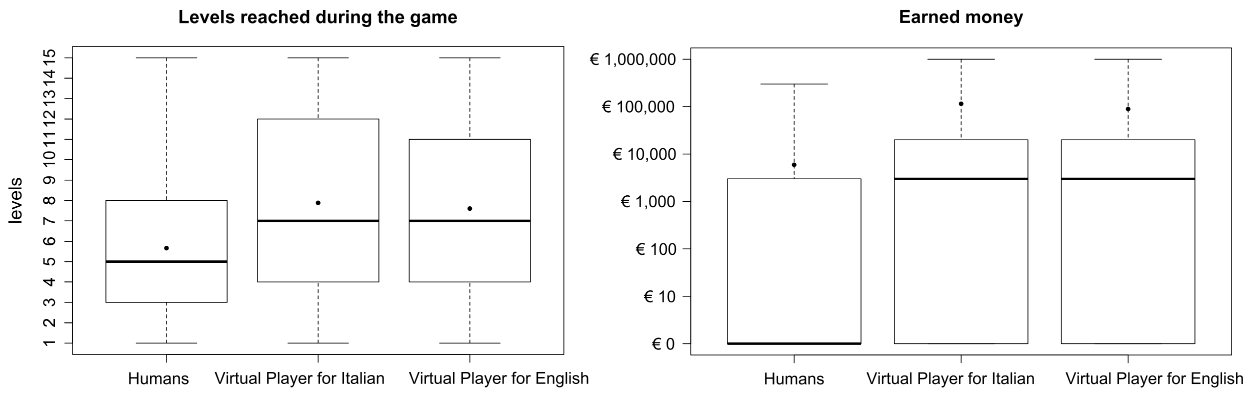


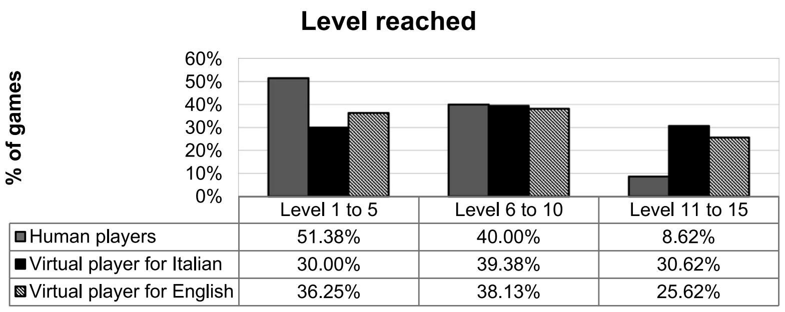
Fig. 11. Distribution of the levels reached during the game and the money earned by the players (in log scale). Upper and lower ends of the boxes represent the 3rd and 1st quartile, respectively. Whiskers extend to the most extreme data point which is no more than 1.5 times the interquartile range. Median values are depicted with solid lines, mean values with solid points.

We involved 35 subjects, different from those involved in the previous experiment, who overall played 325 games. Each game consists of questions selected by the Game Manager, which randomly sampled one question for each level as the game proceeds (games played by the same human player always have different questions for the same level). On the other side, the virtual player played 160 games, whose questions are also selected by the Game Manager from both the Italian and English datasets. An important setting to configure the virtual player is the value of the *threshold* used by the decision making algorithm for assessing a situation of uncertainty (difference between the score of the best and the second best candidate answer – see Section 7). The *threshold* value was optimized by empirically varying its value, and selecting the one which led the virtual player to obtain the highest average income on the questions in the validation set of the first fold. *Threshold* was finally set to 0.2 for both Italian and English (plot showing how different values affect the average income depicted in Fig. 10).

Fig. 11 shows the boxplots of the levels reached during the game and of the money earned by the players, while Figs. 12 and 13 report the distribution of games reaching a specific level and the distribution of games ended with the income in a specific interval.

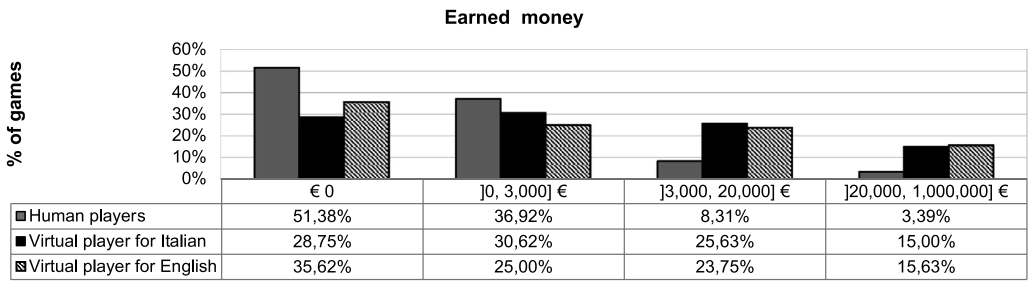
All the players are able to reach the last level of the game, but the average level reached by humans is between five and six (5.65), with respect to the virtual player which reaches a level between seven and eight, i.e. 7.88 and 7.60 for Italian and English, respectively. This is also highlighted by the median value, which is the fifth level for humans and the seventh for the virtual player. More than half of the times (51.38%) the human players ended the game by reaching levels from 1 to 5, while 40% of times reached levels 6 to 10. Few times (8.62%) humans were able to reach the last levels (level 15 reached only once). The distribution of the games ended by the virtual player in the three groups of questions is almost uniform for both Italian and English (with a slightly better performance for Italian), and this is coherent with the results in Figs. 8 and 9, in which the accuracy of the system is very similar for all the levels of the game.

301 out of 325 games (92.61%) played by humans ended due to an error in the response, while 24 times (7.39%) the players ended the game by retiring and taking the earned money (in three cases the players retired from the game at level 6 and 11, even though a wrong answer would not have any effect on the earned money). Moreover, human players never ended the game with the maximum prize. Differently from humans, the virtual player was able to successfully complete the game. Indeed, it earned € 1,000,000 17 times (10.62%) for Italian, and 12 times (7.50%) for English. 116 games (72.50%) ended due to a wrong answer by the virtual player for Italian, while 27 times (16.87%) it retired from the game without



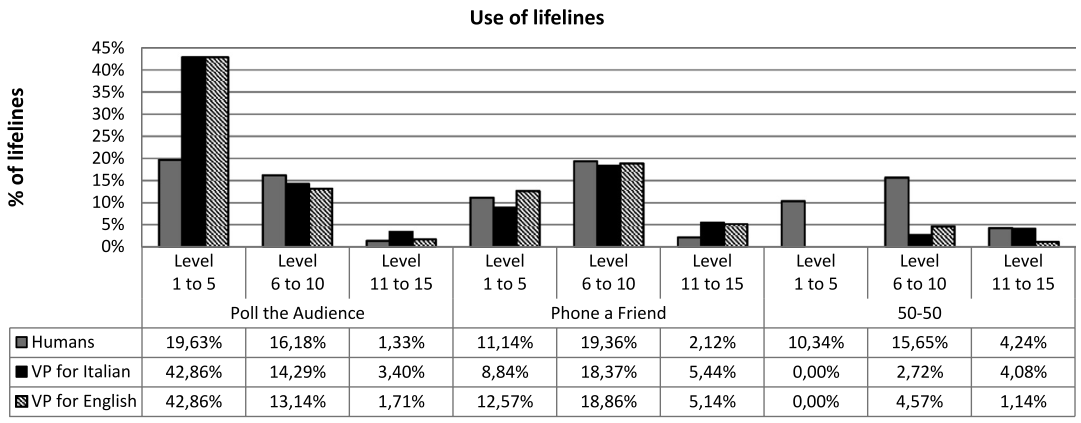
	Level 1 to 5	Level 6 to 10	Level 11 to 15
▩ Human players	51.38%	40.00%	8.62%
■ Virtual player for Italian	30.00%	39.38%	30.62%
▧ Virtual player for English	36.25%	38.13%	25.62%

Fig. 12. Distribution of games reaching a specific level.



	€ 0	]0, 3,000] €	]3,000, 20,000] €	]20,000, 1,000,000] €
▩ Human players	51,38%	36,92%	8,31%	3,39%
■ Virtual player for Italian	28,75%	30,62%	25,63%	15,00%
▧ Virtual player for English	35,62%	25,00%	23,75%	15,63%

Fig. 13. Distribution of games ended with the income in a specific interval.



	Poll the Audience			Phone a Friend			50-50		
	Level 1 to 5	Level 6 to 10	Level 11 to 15	Level 1 to 5	Level 6 to 10	Level 11 to 15	Level 1 to 5	Level 6 to 10	Level 11 to 15
▩ Humans	19,63%	16,18%	1,33%	11,14%	19,36%	2,12%	10,34%	15,65%	4,24%
■ VP for Italian	42,86%	14,29%	3,40%	8,84%	18,37%	5,44%	0,00%	2,72%	4,08%
▧ VP for English	42,86%	13,14%	1,71%	12,57%	18,86%	5,14%	0,00%	4,57%	1,14%

Fig. 14. Distribution of the lifelines used during the game.

providing the answer. The virtual player for English ended 114 games (71.25%) due to a wrong answer, and retired from the game 34 times (21.25%). The highest percentage of games ended by the virtual player without providing the answer highlights its more conservative and low risk strategy. It is interesting to note that the decision making algorithm does not allow the virtual player to end the game at level 6 or 11.

The money earned by humans is € 5926 on average, while the average income of the virtual player is significantly higher. Indeed, it earned € 114,531 for Italian, and € 88,878 for English. The detailed figures are shown in Fig. 13, where the games ended with a null income are reported, as well as the games ended in each interval corresponding to milestone questions. Most of the games played by humans (88.30%) ended with a null income or by answering questions in the first group. This means that they reached the last two groups of questions 11.70% of times, differently from the virtual player which reached the last two groups of questions about 40% of times for both Italian and English. The fewer percentage of games ended with a null income by the virtual player confirms its risk averse behavior (albeit some differences between Italian and English exist) that, coupled with its ability to provide correct answers regardless the level of the game, allows it to end the games with a higher average income.

Fig. 14 reports the use of the lifelines during the different stages of the game. The comparison of the strategies adopted by humans and the virtual player is not fair since the latter uses a static order for the available lifelines, as defined by the decision making algorithm (i.e. *Poll the Audience, Phone a Friend, 50:50*). Despite this limitation, overall, the support provided by the lifelines to the virtual player is valuable. Indeed, the virtual player is able to provide the correct answer 114 times for Italian and 141 times for English thanks to the lifelines. Human players and the virtual player use a single lifeline per game on average, hence they behave in a very similar way. During the first stages of the game, the frequency of usage of *Poll the Audience* by the virtual player (42.86% both for Italian and for English) is as much as the cumulative frequency of use of all the three available lifelines by the human players (41.11%). This means that the virtual player is forced to use that lifeline before the others, as defined in the decision making algorithm, while the human players use all the available lifelines, even though they prefer *Poll the Audience* during the first stages of the game. The frequency of use of *Phone a Friend* by humans and the virtual player is quite similar. As expected, *50:50* is never used by the virtual player during the first levels of the game, while humans uniformly used it in all the levels of the game.

To sum up, as expected from the results of Experiment 1, the virtual player is able to outperform humans, even though it adopts a very simple decision making strategy. The better performance is in terms of both the average reached level and the earned money at the end of the game. The performance of the virtual player for Italian and English is pretty much the same, except the average income, which is significantly better for Italian than English. As observed for the performance of the QA and Answer Scoring, this is likely due to the different size of the knowledge sources for Italian and for English, as well as of course to the different datasets used in the experiment.

9. Conclusions

In this work we investigated two issues:

- To what extent can a QA system be designed in a language-independent way, by preserving its effectiveness?
- Can Wikipedia and DBpedia serve as effective knowledge bases for answering WWBM game questions?

As regards the first issue, this work actually led to the definition of an effective language-independent framework for QA, in which both the Question Answering and Answer Scoring modules were defined using a set of filters that are not related to a specific language. We also defined an effective strategy to combine different criteria for scoring candidate answers through machine learning techniques. Experiments performed on Italian and English showed the effectiveness of the approach.

As regards the second issue, we built a virtual player for the WWBM game based on the following modules:

- Question Answering: is able to retrieve passages of text relevant to a specific question expressed in natural language, by using Wikipedia and DBpedia open knowledge sources;
- Answer Scoring: implements several heuristics based on the analysis of the results returned by the Question Answering module, in order to assign a score to the four candidate answers;
- Decision Making: chooses the strategy to play the game, by exploiting the scores of the four candidate answers, the availability of lifelines, and the current level of the game.

The virtual player outperformed human players in terms of average accuracy in correctly answering to questions of the game, and in terms of ability to play real games with their rules.

We plan to enhance the decision making algorithm, in order to allow a smarter management of the lifelines, and a less conservative strategy which could lead to a more risk neutral player, with the hope that these refinements will further improve the overall accuracy of the system. Moreover, specific heuristics could be devised to answer to questions which are currently unanswerable for our system (Section 8.1.1) in order to further improve its performance.

References

[1] N. Aggarwal, P. Buitelaar, A system description of natural language query over DBpedia, in: C. Unger, P. Cimiano, V. Lopez, E. Motta, P. Buitelaar, R. Cyganiak (Eds.), Proceedings of Interacting with Linked Data (ILD 2012), Workshop Co-Located with ESWC 2012, in: CEUR Workshop Proceedings, vol. 913, 2012, pp. 96–99.

[2] K. Ahn, J. Bos, D. Kor, M. Nissim, B.L. Webber, J.R. Curran, Question answering with QED at TREC 2005, in: E.M. Voorhees, L.P. Buckland (Eds.), Proceedings of the Fourteenth Text REtrieval Conference, TREC 2005, in: Special Publications, vol. 500-266, National Institute of Standards and Technology (NIST), 2005.

[3] P. Basile, M. de Gemmis, P. Lops, G. Semeraro, Solving a complex language game by using knowledge-based word associations discovery, IEEE Trans. Comput. Intell. AI Games (2015), http://dx.doi.org/10.1109/TCIAIG.2014.2355859, in press.

[4] J. Berant, A. Chou, R. Frostig, P. Liang, Semantic parsing on freebase from question–answer pairs, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18–21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, ACL, 2013, pp. 1533–1544, a meeting of SIGDAT, a special interest group of the ACL.

[5] C. Bizer, The emerging Web of linked data, IEEE Intell. Syst. 24 (2009) 87–92.

[6] E. Breck, J.D. Burger, L. Ferro, L. Hirschman, D. House, M. Light, I. Mani, How to evaluate your question answering system every day and still get real work done, in: Proceedings of the Second International Conference on Language Resources and Evaluation, LREC 2000, European Language Resources Association, 2000, pp. 1495–1500.

[7] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.

[8] E. Cabrio, A.P. Aprosio, J. Cojan, B. Magnini, F. Gandon, A. Lavelli, QAKiS@QALD-2, in: C. Unger, P. Cimiano, V. Lopez, E. Motta, P. Buitelaar, R. Cyganiak (Eds.), Proceedings of Interacting with Linked Data (ILD 2012), Workshop Co-Located with EWSC 2012, in: CEUR Workshop Proceedings, vol. 913, 2012, pp. 87–95.

[9] J.J. Castillo, The contribution of FaMAF at QA@CLEF 2008: answer validation exercise, in: Working Notes of the CLEF 2008 Workshop, 2008, pp. 17–19.

[10] J. Chen, A. Diekema, M.D. Taffet, N.J. McCracken, N.E. Ozgencil, O. Yilmazel, E.D. Liddy, Question answering: CNLP at the TREC-10 question answering track, in: Text Retrieval Conference, in: Special Publications, vol. 500-274, National Institute of Standards and Technology (NIST), 2001.

[11] P. Cimiano, M. Minock, Natural language interfaces: what is the problem? – a data-driven quantitative analysis, in: H. Horacek, E. Métais, R. Muñoz, M. Wolska (Eds.), Natural Language Processing and Information Systems, 14th International Conference on Applications of Natural Language to Information Systems, NLDB 2009, in: Lecture Notes in Computer Science, vol. 5723, Springer, 2010, pp. 192–206, revised papers.

[12] I. Dagan, O. Glickman, B. Magnini, The PASCAL recognising textual entailment challenge, in: J.Q. Candela, I. Dagan, B. Magnini, F. d'Alché Buc (Eds.), Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop, in: Lecture Notes in Computer Science, vol. 3944, Springer, 2005, revised selected papers.

[13] D. Damljanovic, M. Agatonovic, H. Cunningham, FREyA: an interactive way of querying linked data using natural language, in: R. Garcia-Castro, D. Fensel, G. Antoniou (Eds.), The Semantic Web: ESWC 2011 Workshops, in: Lecture Notes in Computer Science, vol. 7117, Springer, 2012, pp. 125–138, revised selected papers.

[14] S. Dumais, M. Banko, E. Brill, J. Lin, A. Ng, Web question answering: is more always better?, in: Proceedings of the 25th ACM SIGIR Conference, SIGIR'02, ACM, New York, NY, USA, 2002, pp. 291–298.

[15] M. Ernandes, G. Angelini, M. Gori, WebCrow: a Web-based system for crossword solving, in: M.M. Veloso, S. Kambhampati (Eds.), AAAI, AAAI Press/The MIT Press, 2005, pp. 1412–1417.

[16] A. Fader, L. Zettlemoyer, O. Etzioni, Open question answering over curated and extracted knowledge bases, in: S.A. Macskassy, C. Perlich, J. Leskovec, W. Wang, R. Ghani (Eds.), The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014, ACM, 2014, pp. 1156–1165.

[17] D. Ferrucci, A. Levas, S. Bagchi, D. Gondek, E.T. Mueller, Watson: beyond Jeopardy!, Artif. Intell. (2013) 93–105.

[18] D.A. Ferrucci, E.W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J.W. Murdock, E. Nyberg, J.M. Prager, N. Schlaefer, C.A. Welty, Building Watson: an overview of the DeepQA project, AI Mag. 31 (2010) 59–79.

[19] E. Gabrilovich, S. Markovitch, Computing semantic relatedness using Wikipedia-based explicit semantic analysis, in: M.M. Veloso (Ed.), Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007, Hyderabad, India, January 6–12, 2007, Morgan Kaufmann, 2007, pp. 1606–1611.

[20] I. Glöckner, University of Hagen at QA@CLEF: answer validation exercise, in: Working Notes of the CLEF 2008 Workshop, 2008, pp. 17–19.

[21] S.M. Harabagiu, S. Maiorano, Finding answers in large collections of texts: paragraph indexing + abductive inference, in: Proceedings of the AAAI Fall Symposium on Question Answering, AAAI, 1999, pp. 63–71.

[22] S.M. Harabagiu, D.I. Moldovan, M. Paşca, R. Mihalcea, M. Surdeanu, R.C. Bunescu, R. Girju, V. Rus, P. Morarescu, FALCON: boosting knowledge for answer engines, in: Text Retrieval Conference, in: Special Publications, vol. 500-249, National Institute of Standards and Technology (NIST), 2000, pp. 479–488.

[23] S.M. Harabagiu, M. Paşca, S.J. Maiorano, Experiments with open-domain textual question answering, in: Proceedings of the 18th Conference on Computational Linguistics – Volume 1, COLING'00, Association for Computational Linguistics, Stroudsburg, PA, USA, 2000, pp. 292–298.

[24] E.H. Hovy, L. Gerber, U. Hermjakob, M. Junk, C.Y. Lin, Question answering in Webclopedia, in: Text Retrieval Conference, in: Special Publications, vol. 500-249, National Institute of Standards and Technology (NIST), 2000, pp. 655–664.

[25] S.K. Lam, D.M. Pennock, D. Cosley, S. Lawrence, 1 billion pages = 1 million dollars? Mining the Web to play "Who Wants to Be a Millionaire?", in: C. Meek, U. Kjærulff (Eds.), Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence, Morgan Kaufmann, 2003, pp. 337–345.

[26] J.J. Lin, An exploration of the principles underlying redundancy-based factoid question answering, ACM Trans. Inf. Syst. 25 (2007).

[27] M.L. Littman, Review: computer language games, in: T.A. Marsland, I. Frank (Eds.), Computers and Games, 2nd Int. Conf., in: Lecture Notes in Computer Science, vol. 2063, Springer, 2000, pp. 396–404, revised papers.

[28] M.L. Littman, G.A. Keim, N. Shazeer, A probabilistic approach to solving crossword puzzles, Artif. Intell. 134 (2002) 23–55.

[29] T. Liu, Learning to Rank for Information Retrieval, Springer, 2011.

[30] V. Lopez, M. Fernández, E. Motta, N. Stieler, PowerAqua: supporting users in querying and exploring the Semantic Web, Semant. Web 3 (2012) 249–265.

[31] V. Lopez, C. Unger, P. Cimiano, E. Motta, Evaluating question answering over linked data, J. Web Semant. 21 (2013) 3–13.

[32] B. Magnini, M. Negri, R. Prevete, H. Tanev, Is it the right answer? Exploiting Web redundancy for answer validation, in: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, ACL, 2002, pp. 425–432.

[33] C.D. Manning, P. Raghavan, H. Schtze, Introduction to Information Retrieval, Cambridge University Press, New York, NY, USA, 2008.

[34] M. Maybury, O. Stock, W. Wahlster, Intelligent interactive entertainment grand challenges, IEEE Intell. Syst. 21 (2006) 14–18.

[35] P. Molino, P. Basile, QuestionCube: a framework for question answering, in: G. Amati, C. Carpineto, G. Semeraro (Eds.), Proceedings of the 3rd Italian Information Retrieval (IIR) Workshop, Bari, Italy, January 26–27, 2012, in: CEUR Workshop Proceedings, vol. 835, CEUR-WS.org, 2012, pp. 167–178.

[36] P. Molino, P. Basile, A. Caputo, P. Lops, G. Semeraro, Exploiting distributional semantic models in question answering, in: IEEE International Conference on Semantic Computing, IEEE, 2012, pp. 146–153.

[37] P. Molino, P. Basile, C. Santoro, P. Lops, M. de Gemmis, G. Semeraro, A virtual player for "Who Wants to Be a Millionaire?" based on question answering, in: M. Baldoni, C. Baroglio, G. Boella, R. Micalizio (Eds.), AI*IA 2013: Advances in Artificial Intelligence – XIIIth International Conference of the Italian Association for Artificial Intelligence, in: Lecture Notes in Computer Science, vol. 8249, Springer, 2013, pp. 205–216.

[38] C. Monz, Minimal span weighting retrieval for question answering, in: R. Gaizauskas, M. Greenwood, M. Hepple (Eds.), Proceedings of the SIGIR 2004 Workshop on Information Retrieval for Question Answering, pp. 23–30.

[39] M. Paşca, Open-Domain Question Answering from Large Text Collections, Studies in Computational Linguistics, CSLI Publications, 2003.

[40] A. Peñas, E.H. Hovy, P. Forner, Á. Rodrigo, R.F.E. Sutcliffe, R. Morante, QA4MRE 2011–2013: overview of question answering for machine reading evaluation, in: P. Forner, H. Müller, R. Paredes, P. Rosso, B. Stein (Eds.), Information Access Evaluation. Multilinguality, Multimodality, and Visualization – 4th International Conference of the CLEF Initiative, CLEF 2013, in: Lecture Notes in Computer Science, vol. 8138, Springer, 2013, pp. 303–320.

[41] F. Perea, J. Puerto, Dynamic programming analysis of the TV game "Who Wants to Be a Millionaire?", Eur. J. Oper. Res. 183 (2007) 805–811.

[42] S. Robertson, H. Zaragoza, The probabilistic relevance framework: BM25 and beyond, Found. Trends Inf. Retr. 3 (2009) 333–389.

[43] J. Rocchio, Relevance feedback information retrieval, in: G. Salton (Ed.), The SMART Retrieval System – Experiments in Automated Document Processing, Prentice-Hall, Englewood Cliffs, NJ, 1971, pp. 313–323.

[44] Á. Rodrigo, A. Peñas, F. Verdejo, UNED at answer validation exercise 2007, in: C. Peters, V. Jijkoun, T. Mandl, H. Müller, D.W. Oard, A. Peñas, V. Petras, D. Santos (Eds.), Advances in Multilingual and Multimodal Information Retrieval, 8th Workshop of the Cross-Language Evaluation Forum, CLEF 2007, in: Lecture Notes in Computer Science, vol. 5152, Springer, 2008, pp. 404–409, revised selected papers.

[45] Á. Rodrigo, A. Peñas, F. Verdejo, Overview of the answer validation exercise 2008, in: C. Peters, T. Deselaers, N. Ferro, J. Gonzalo, G.J.F. Jones, M. Kurimo, T. Mandl, A. Peñas, V. Petras (Eds.), Evaluating Systems for Multilingual and Multimodal Information Access, 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008, in: Lecture Notes in Computer Science, vol. 5706, Springer, 2009, pp. 296–313, revised selected papers.

[46] S. Royce, B. Straits, Approaches to Social Research, 3rd edition, Oxford University Press, New York, 1999.

[47] M. Sahlgren, An introduction to random indexing, in: Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, 2005.

[48] G. Semeraro, M. de Gemmis, P. Lops, P. Basile, An artificial player for a language game, IEEE Intell. Syst. 27 (2012) 36–43.
[49] J.A. Shaw, E.A. Fox, Combination of multiple searches, in: Proceedings of the Second Text REtrieval Conference, TREC-2, in: Special Publications, vol. 500-215, National Institute of Standards and Technology (NIST), 1994, pp. 243–252.
[50] A. Singhal, C. Buckley, M. Mitra, Pivoted document length normalization, in: Proceedings of the 19th ACM SIGIR Conference, SIGIR'96, ACM, New York, NY, USA, 1996, pp. 21–29.
[51] P. Smolensky, Tensor product variable binding and the representation of symbolic structures in connectionist systems, Artif. Intell. 46 (1990) 159–216.