QuestionCube: a framework for Question Answering

Piero Molino and Pierpaolo Basile

QuestionCube s.r.l.
{piero.molino,pierpaolo.basile}@questioncube.com
www.questioncube.com

Abstract. QuestionCube is a framework for Question Answering (QA) that combines several techniques to retrieve passages containing the exact answers for natural language questions. It exploits: (a) Natural Language Processing algorithms for question and candidate answers analysis both in English and Italian; (b) Information Retrieval probabilistic models for candidate answers retrieval and (c) Machine Learning methods for question classification. The data source for the answer is an unstructured text document collection stored in search indices. In this paper an overview of the QuestionCube framework architecture is provided, together with a description of Wikiedi, a QA system for Wikipedia which exploits the proposed framework.

1 Introduction

Question Answering (QA) emerged in the last decade as one of the most promising fields in Artificial Intelligence due to some competitions organized during international conferences [31, 25], but the first studies can be dated back to 1960s [3, 29]. In the last years some enterprise applications shown the potential of the state of the art technology, for example the IBM's Watson/DeepQA system [12, 11]. By exploiting techniques borrowed from Information Retrieval and Natural Language Processing (NLP), QA systems are able to answer user questions expressed in natural language with short passages of text which contain the exact answer or sometimes directly with the exact answer, depending on the domain, rather than returning long lists of full-text documents that users have to check in order to find the information needed, as most search engines do.

Most closed-domain QA systems use a variety of NLP methods to help the understanding of user's queries and the matching of passages extracted from documents [13, 15, 7]. The most commonly adopted linguistic analysis steps include: stemming, lemmatization with dictionaries, part-of-speech tagging, parsing, named entity recognition, lexical semantics (Word Sense Disambiguation), etc. The use of those NLP steps is fundamental to find the correct answer in closed-domain QA, since there is likely to be few answers to any user's question and the way in which they are expressed may be significatively different from the question. The difficulty of the task lies in mapping questions to answers by way of uncovering complex lexical, syntactic, or semantic relationships between questions and candidate answers.

Open-domain QA systems, instead, have to face different types of problems: the probability of finding correct answers is higher, but the noise produced from the Web is also much higher than in the case of closed domain. Most systems exploit redundancy and textual pattern extraction and matching to solve the problem [9, 14, 24, 19].

The main limitation of current systems working on specific document collections is that they focus on precise tasks and are not general enough. On the other hand, open-domain systems, particularly those working on the World Wide Web, have long response times and lack in accuracy.

This paper describes QuestionCube, a framework for building QA systems with focus on closed domains, but which could be easily applied to open domains as well. It exploits NLP algorithms for both English and Italian and integrates a question categorization component based on Machine Learning techniques and linguistic rules written by human experts. Text document collections used as data sources are organized in indices for generic unstructured data storage with fast and reliable search functions exploiting state-of-the-art Information Retrieval weighting schemes.

The paper is structured as follows. Section 2 provides a generic overview of the framework architecture, while in Section 3 details about main components for analysis, search and filtering are described. Section 4 presents Wikiedi, a proof-of-concept system which relies on the QuestionCube framework and exploits Wikipedia pages as data source. Final conclusions, then, close the paper.

2 Framework overview

QuestionCube is a multilingual QA framework built using NLP and IR techniques.

The architecture, shown in Figure 1, is similar to the one proposed in [27], but it differs in several important aspects that make it more general and easier to expand. The first step is a linguistic analysis of the user's question. Question analysis is performed by a pipeline of NLP analyzer. The NLP components tag the question at different linguistic levels. The linguistic tagging process allows to classify the question according to a shared question-type hierarchy. The question classifier uses an ensemble learning approach that exploits both hand-written rules and rules inferred by machine learning categorization techniques, thus bringing together the hand-written rules' effectiveness and precision and the machine learning classifier's recall. The question is then passed to the search engines, whose architecture is highly parallel and distributed. Moreover, each single engine has its own query generator, because query's structure and syntax may change across different engines. The filter pipeline is then responsible for the scoring and the filtering of the passages retrieved by the search engines. Finally, the ranked list of passages is presented to the user.

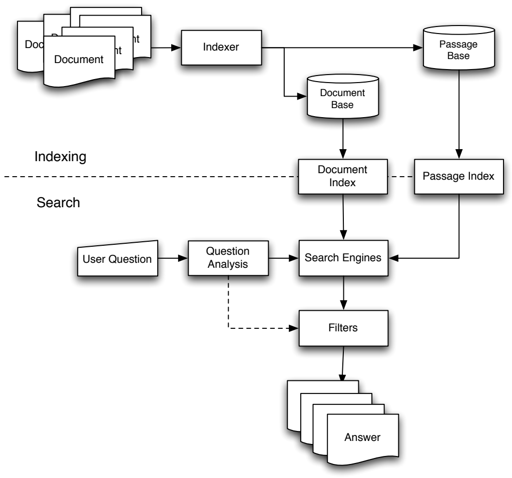


Fig. 1. QuestionCube architecture overview

The main motivation behind QuestionCube architecture is to create a Question Answering system simply by the dynamic composition of framework components. The high level of abstraction of the components allows to add support to a new language by just creating new interchangeable analyzers which implement the algorithms for the specific language. Another point that must be underlined is that our approach relies on several search engines in order to exploit different data source. For example, documents and passages could be retrieved from a database, from a Web search engine or from an enterprise search engine. The parallel approach allows to query several data sources at the same time.

3 Details

3.1 Question Analysis

The macro-component of the question analysis is composed of a pipeline of NLP analyzers, a data-structure to represent linguistic annotated text and the question classifier, as shown in Figure 2.

The NLP pipeline is easily configurable depending on the application domain of the QA system. Obviously, a small number of basic NLP analyzers added to the pipeline allows faster tagging, while more components in the pipeline requires more time for deeper linguistic analysis.

NLP analyzers are provided for both English and Italian. The stemmer is implemented by Snowball[1] both for English and Italian. The lemmatization is realized exploiting the morpho-syntactic analyzer of the WordNet API [10] for

[1] Available on-line: http://snowball.tartarus.org/

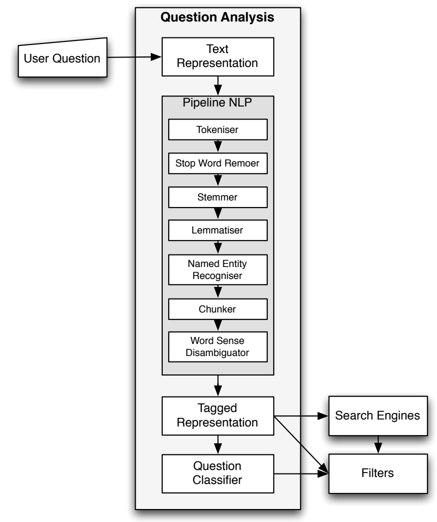


Fig. 2. Question analysis macro-component

the English, while Morph-it [32] is exploited for the Italian. Named Entity Recognition (NER) is performed by a machine learning classifier based on Support Vector Machines [8] using an open-source tool called YAMCHA [16]. The same tool is used for the chunker component. Both in chunking and NER, POS-tags and lemmas are adopted as features. The Word Sense Disambiguation (WSD) is implemented by the UKB algorithm [1], which is a graph-based technique based on a personalized version of PageRank [4] over WordNet graph.

The output of the NLP analyzers is a set of tags that are added to the text representation. The text representation is the input for the search engines, for the classifier and also for the filters, as they need linguistic information about the question to match it with the answers.

The NLP pipeline is also used by each filter to analyze the candidate answer at the same linguistic level as the question.

3.2 Question Classifier

The annotated text representation of the question is used by the question classifier. It is composed by three classifiers as shown in Figure 3.

The first one is based on Support Vector Machines and uses the tags from the text representation as features to classify the question. The main features are the head word of the question, the terms, their PoS tags, semantic identifiers provided by WSD and Named Entities.

The other two classifiers are rule-based ones that exploit respectively hand-written and learned rules in the form of regular expressions based on Named Entity categories and semantic identifiers.

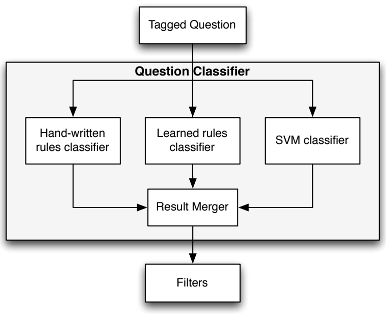


Fig. 3. Question classifier macro-component

The outputs of the classifiers are merged by using a weighted voting system that returns a question category.

The category is selected among the ones in the typology proposed in [17, 18]. Categories are exploited by filters in order to give a higher score to those candidate answers containing Named Entities in accordance with the question category.

3.3 Search Engine

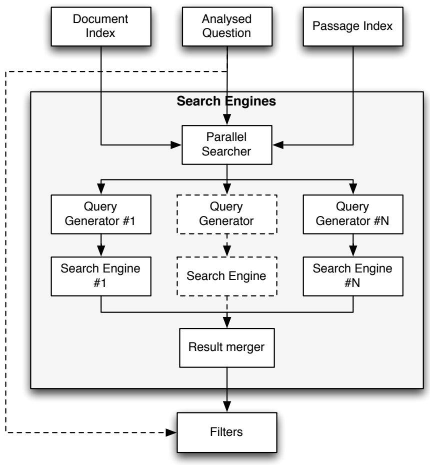


Fig. 4. Search engine macro-component

The search engine macro-component is designed to work in parallel and distributed environment. It allows to implement several information retrieval strategies and thus to aggregate their results, as shown in Figure 4.

The parallel engine is modular and it is possible to add an arbitrary number of different search engines inside it. It calls each engine when a new question comes and merges their outputs in a single list. The list contains all the candidate answers from all the engines, each one with a reference to the engines that retrieved it and the score assigned by each engine. Some filters normalize those scores in order to get an overall best score. Each single search engine has its own query generation component, because the syntax of the query may change among different engines. Each query generator may use different annotations from the text representation: some may use only tokens, others can use lemmas or stems, others may use WordNet synsets to generate the query. This approach allows to add a new search engine inside the framework with minimal effort. The main goal of using more than one search engine is to rely on different retrieval strategies in order to take the best results from each one. For example, in the current implementation, we adopt two search engines: the first one works on keywords, while the second one relies on lemmas. Moreover, the use of multiple search engines allows to use different retrieval models merging the results in an unique result set.

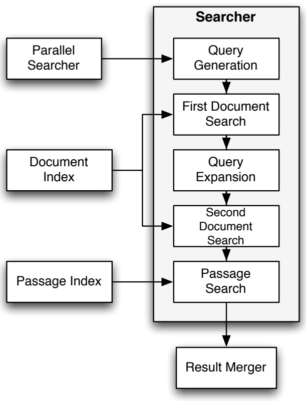


Fig. 5. Single search engine

The process performed by each search engine is described in Figure 5. The query generator builds the query for its search engine from the text representation provided by the parallel engine. Each query generator may implement different query improvement techniques (such as relevance feedback and query expansion). The query is executed by the search engine that returns the best scoring documents. The passage index is used to obtain the passages from retrieved documents. These passages are merged into one single list by an aggregation component and then passed to the filters which score, sort and filter them.

The QuestionCube framework provides a search engine based on *BM25* model [26]. The query generation component for this searcher allows three different query improvement techniques:

- *Query expansion* through WordNet synonyms of the synsets found in the question;
- *Kullback-Liebler Divergence*, a statistical technique that exploits the terms distribution of the top-ranked documents [6, 20];
- *Divergence From Randomness*, a statistical technique that weights the terms distribution with the *Bo1* weighting scheme [2].

It is important to underline that the WordNet based query expansion is used only if the question has been disambiguated.

3.4 Filters

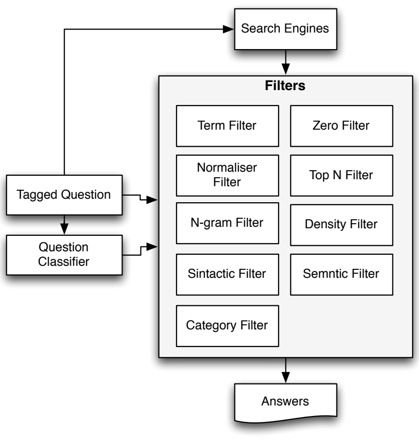


Fig. 6. Candidate answers filtering macro-component

This macro-component, sketched in Figure 6, contains all the passages filters. It allows to build a pipeline in which it is possible to add filters. If there is no dependence between the filters, it is possible to place them in any order to create different pipelines for several domains and needs.

Each filter checks every passage in input obtained from the search engine and assigns a score to them depending on the implemented logic. Each filter can exploit information provided by the text representation and use the category tag assigned to the question by the classifier. Some filters do not assign scores but just sort the passages according to some score or ranking threshold. The composition of the filters in the pipeline is important to determine the quality of the results returned by the system, its efficiency and the time taken to give an answer.

A description of the logic of each filter is given below:

- **Zero Filter**: removes from the list all those passages that, at the moment of the analysis, have a general score of 0;
- **Top-N Filter**: sorts passages in a decreasing order according to their current score and removes all those passages under the N-th position in the ranking (N is given as input to the filter);
- **Terms filter**: assigns a score to every analyzed passage based on the frequency of the question terms in the passage;
- **Normalization Filter**: assigns a score to each analyzed passage based on the passage length, by normalizing its overall score. Both a simple normalization filter (which considers only the number of terms and is generally called *Byte-size Normalization*) and a filter based on the *Pivoted Normalised Document Length* technique are implemented. Both techniques and their effectiveness are discussed in [21, 30];
- **N-grams Filter**: assigns a score to each analyzed passage based on the overlapping of n-grams between the question and the passage (n is given as input to the filter);
- **Density Filter**: assigns a score to each analyzed passage based on the distance of the question terms inside the passage increasing the score of those passages in which the question terms are closer. The density is calculated by the Minimal Span Weighting schema proposed by [22]:
$\left(\frac{|q \cap d|}{1+\max(mms)-\min(mms)}\right)^{\alpha}\left(\frac{|q \cap d|}{|q|}\right)^{\beta}$
where q and d are the set of terms respectively of the query and the document (specifically here, 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; and α and β are two parameters.
- **Syntactical Filter**: assigns a score to each analyzed passage based on the Phrase Matching algorithm, presented in [23]. The algorithm takes into account the head of each phrase. If the head is common to the two considered texts (in this case the query and the passage), the maximal overlapping length of each phrase is calculated.
- **Semantic Filter**: assigns a score to each analyzed passage based on the frequency of terms tagged with the same WordNet synsets inside both question and passage. A more complex filter that calculates a semantic similarity measure between texts based on the semantic distance measure described in [5] is one of the future developments;
- **Category Filter**: assigns a score to each analyzed passage based on a list of pairs that link the question categories to typologies of named entity: if, on the basis of the question category, entities of the expected typology are found in the passage the score will be positive.

– **Z-Score Filter**: assigns a score to each analyzed passage based on the Z-Score normalization [28] of scores assigned by search engines and other filters.

A boost factor can be assigned to each filter which intensifies or decreases its strength.

4 Wikiedi

Wikiedi is a Web application that allows users to ask questions and receive answers extracted from articles from Italian and English Wikipedia. The Question Answering core of Wikiedi is built on the QuestionCube framework with a specific configuration that balances accuracy and reactivity.

The system is configured to index Wikipedia pages with their respective linguistic annotations. This ensures quick response time because NLP algorithms will not process linguistically each passage at runtime. To improve performances, the annotated passages are represented in a compact binary structure stored in a database. This allows fast passage retrieval reducing to zero the reconstruction time.

The filters adopted in Wikiedi range from the most basic ones that work on tokens to the most sophisticated ones exploiting semantics.

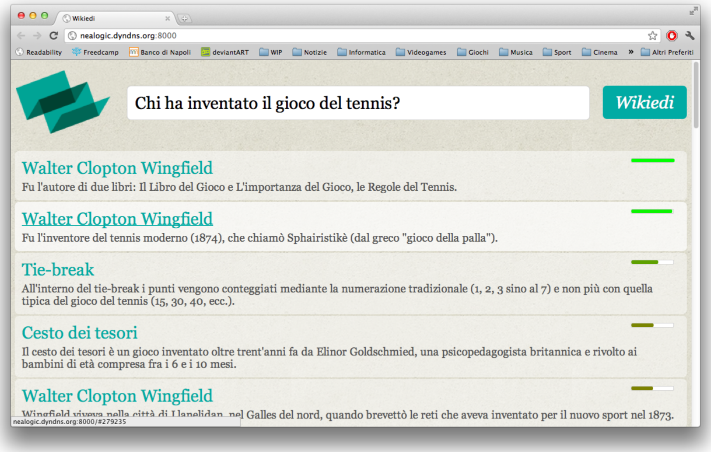
The decision to use documents from Wikipedia to evaluate the potential of QuestionCube framework is motivated by the heterogeneous nature of the information on Wikipedia. This reflects the enterprise context where documents that belong to different domains are stored in a single collection increasing the noise in the retrieval phase.

The other goal of Wikiedi is to engage the user in improving system performances. After the user has submitted a question to the system, Wikiedi will display an ordered list of answers. The user will have the possibility of voting for the correct answer, so that the system can use the feedback to improve precision and recall in future queries. The next time the question is issued, the results will be sorted by mixing the score given by the system and users' judgements.

Moreover, when one of the answers provided by Wikiedi is not correct, users will have the opportunity of inserting the correct one. Using this strategy it is possible to enrich the system with additional information. Users asking the same question will then obtain both automatically obtained results alongside with user added answers.

To meet users information needs, the QuestionCube framework also allows to implement "similar questions" function easily by indexing user questions as they are asked and calculating their similarity. Moreover, the framework allows to implement a simple content-based recommender system that suggests questions the user may be also interested in.

The results are shown segment by segment, as shown in Figure 7. Clicking on a result, a page of the full Wikipedia article text is shown. The page is automatically enriched mashing up several multimedia contents from Web 2.0 websites such as Fotopedia, Flickr, Youtube and Vimeo.



Chi ha inventato il gioco del tennis?

Walter Clopton Wingfield
Fu l'autore di due libri: Il Libro del Gioco e L'importanza del Gioco, le Regole del Tennis.

Walter Clopton Wingfield
Fu l'inventore del tennis moderno (1874), che chiamò Sphairistikè (dal greco "gioco della palla").

Tie-break
All'interno del tie-break i punti vengono conteggiati mediante la numerazione tradizionale (1, 2, 3 sino al 7) e non più con quella tipica del gioco del tennis (15, 30, 40, ecc.).

Cesto dei tesori
Il cesto dei tesori è un gioco inventato oltre trent'anni fa da Elinor Goldschmied, una psicopedagogista britannica e rivolto ai bambini di età compresa fra i 6 e i 10 mesi.

Walter Clopton Wingfield
Wingfield viveva nella città di Llanelidan, nel Galles del nord, quando brevettò le reti che aveva inventato per il nuovo sport nel 1873.

Fig. 7. Wikiedi web interface

The Italian version of Wikiedi is available on-line: www.wikiedi.it. The English version will follow soon on www.wikiedi.com.

As for the evaluation, currently statistics about Wikiedi performances are currently not available, since a large number of users' feedback is needed to evaluate them. However, an evaluation of a system built with the QuestionCube framework has been performed using a standard dataset adopted in QA called CLEF 2010 ResPubliQA [25] based on multi-lingual documents from European Legislation. The dataset consists of 10,855 documents and 200 questions. The system is evaluated using the c@1 measure, which takes into account the accuracy on the first returned passage. Table 1 reports the results of our system for each language. The last column shows the results obtained by the best participant system. The obtained results show improvements both in English and Italian.

Table 1. Results on CLEF 2010 ResPubliQA with c@1 measure

Language	QuestionCube	Best system
Italian	**0.68**	0.63
English	**0.75**	0.73

5 Conclusions

In this paper, the QuestionCube framework has been presented. QuestionCube finds the correct answer to a question by combining Natural Language Processing algorithms, Information Retrieval probabilistic models and Machine Learning methods. Wikiedi was also presented as an example enterprise application. Wikiedi allows the user to ask questions in natural language on Wikipedia pages combining the power of the QuestionCube framework with feedback and additional information provided by the community of users. Finally, an evaluation on a standard dataset, CLEF 2010 ResPubliQA, has been provided, which shows an improvement in comparison to other state-of-the-art systems.

References

1. Agirre, E., Soroa, A.: Personalizing PageRank for word sense disambiguation. In: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics. pp. 33–41. EACL '09, Association for Computational Linguistics, Stroudsburg, PA, USA (2009)
2. Amati, G., Van Rijsbergen, C.J.: Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Trans. Inf. Syst. 20, 357–389 (October 2002)
3. Bert F. Green, J., Wolf, A.K., Chomsky, C., Laughery, K.: Baseball, an automatic question-answerer. Managing Requirements Knowledge, International Workshop on 0, 219 (1961)
4. Brin, S., Page, L.: The Anatomy of a Large-Scale Hypertextual Web Search Engine. In: Seventh International World-Wide Web Conference (WWW 1998) (1998)
5. Budanitsky, A., Hirst, G.: Evaluating WordNet-based Measures of Lexical Semantic Relatedness. Comput. Linguist. 32, 13–47 (March 2006)
6. Carpineto, C., de Mori, R., Romano, G., Bigi, B.: An information-theoretic approach to automatic query expansion. ACM Trans. Inf. Syst. 19, 1–27 (2001)
7. Chen, J., Diekema, A., Taffet, M.D., McCracken, N.J., Ozgencil, N.E., Yilmazel, O., Liddy, E.D.: Question answering: Cnlp at the trec-10 question answering track. In: TREC (2001)
8. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning 20(3), 273–297 (1995)
9. Dumais, S., Banko, M., Brill, E., Lin, J., Ng, A.: Web question answering: is more always better? In: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 291–298. SIGIR '02, ACM, New York, NY, USA (2002)
10. Fellbaum, C.: WordNet: an electronic lexical database. Language, speech, and communication, MIT Press
11. Ferrucci, D.A.: Ibm's watson/deepqa. SIGARCH Computer Architecture News 39(3) (2011)
12. Ferrucci, D.A., Brown, E.W., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A., Lally, A., Murdock, J.W., Nyberg, E., Prager, J.M., Schlaefer, N., Welty, C.A.: Building Watson: An Overview of the DeepQA Project. AI Magazine 31(3), 59–79 (2010)
13. Harabagiu, S.M., Moldovan, D.I., Pasca, M., Mihalcea, R., Surdeanu, M., Bunescu, R.C., Girju, R., Rus, V., Morarescu, P.: Falcon: Boosting knowledge for answer engines. In: TREC (2000)
14. Harabagiu, S.M., Paşca, M.A., Maiorano, S.J.: Experiments with open-domain textual question answering. In: Proceedings of the 18th conference on Computational linguistics - Volume 1. pp. 292–298. COLING '00, Association for Computational Linguistics, Stroudsburg, PA, USA (2000)

15. Hovy, E.H., Gerber, L., Hermjakob, U., Junk, M., Lin, C.Y.: Question answering in webclopedia. In: TREC (2000)
16. Kudo, T., Matsumoto, Y.: Fast Methods for Kernel-Based Text Analysis. In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics. pp. 24–31. ACL, Sapporo, Japan (July 2003)
17. Li, X., Roth, D.: Learning question classifiers. In: Proceedings of the 19th international conference on Computational linguistics - Volume 1. pp. 1–7. COLING '02, Association for Computational Linguistics, Stroudsburg, PA, USA (2002)
18. Li, X., Roth, D.: Learning question classifiers: the role of semantic information. Nat. Lang. Eng. 12, 229–249 (September 2006)
19. Lin, J.: An exploration of the principles underlying redundancy-based factoid question answering. ACM Trans. Inf. Syst. 25 (April 2007)
20. Lv, Y., Zhai, C.: Positional relevance model for pseudo-relevance feedback. In: Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval. pp. 579–586. SIGIR '10, ACM, New York, NY, USA (2010)
21. Manning, C.D., Raghavan, P., Schtze, H.: Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA (2008)
22. Monz, C.: Minimal span weighting retrieval for question answering. In: Gaizauskas, R., Greenwood, M., Hepple, M. (eds.) Proceedings of the SIGIR Workshop on Information Retrieval for Question Answering. pp. 23–30 (2004)
23. Monz, C., de Rijke, M.: Tequesta: The university of amsterdam's textual question answering system. In: TREC (2001)
24. Paşca, M.: Open-domain question answering from large text collections. Studies in computational linguistics, CSLI Publications (2003)
25. Penas, A., Forner, P., Rodrigo, A., Sutcliffe, R.F.E., Forascu, C., Mota, C.: Overview of ResPubliQA 2010: Question Answering Evaluation over European Legislation. In: Braschler, M., Harman, D., Pianta, E. (eds.) Working notes of ResPubliQA 2010 Lab at CLEF 2010 (2010)
26. Robertson, S., Zaragoza, H.: The probabilistic relevance framework: Bm25 and beyond. Found. Trends Inf. Retr. 3, 333–389 (April 2009)
27. Schlaefer, N., Gieselman, P., Sautter, G.: The ephyra qa system at trec 2006. In: TREC (2006)
28. Shaw, J.A., Fox, E.A., Shaw, J.A., Fox, E.A.: Combination of multiple searches. In: The Second Text REtrieval Conference (TREC-2. pp. 243–252 (1994)
29. Simmons, R.: Answering english questions by computer: A survey. Communications of the ACM 8(1), 53–70 (1965)
30. Singhal, A., Buckley, C., Mitra, M.: Pivoted document length normalization. In: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 21–29. SIGIR '96, ACM, New York, NY, USA (1996)
31. Voorhees, E.M., Tice, D.M.: The trec-8 question answering track evaluation. In: In Text Retrieval Conference TREC-8. pp. 83–105 (1999)
32. Zanchetta, E., Baroni, M.: Morph-it! a free corpus-based morphological resource for the italian language. Corpus Linguistics 2005 1(1) (2005)