RQAS: A Rapid QA Scheme with Exponential Elevations of Keyword Rankings

Cheng Zhang and Jie Wang
Department of Computer Science, University of Massachusetts, Lowell, MA 01854, U.S.A.

Keywords: Question Answering, Information Retrieval, Keyword Ranking.

Abstract: We present a rapid question-answering scheme (RQAS) for constructing a chatbot over specific domains with data in the format of question-answer pairs (QAPs). We want RQAS to return the most appropriate answer to a user question in realtime. RQAS is based on TF-IDF scores and exponential elevations of keyword rankings generated by TextRank to overcome the common problems of selecting an answer based on the highest similarity of the user question and questions in the existing QAPs, making it easy for general QAS builders to construct a QAS for a given application. Experiments show that RQAS is both effective and efficient regardless the size of the underlying QAP dataset.

1 INTRODUCTION

Question-answering systems (QAS), a particular type of information retrieving system, must return an appropriate answer in realtime to a question asked by the user. Both questions and answers are expressed in natural languages in the form of text or voice. Since IBM Watson won the "Jeopardy!" champion in 2011 (see https://en.wikipedia.org/wiki/Watson_(computer)), QAS has attracted increasing attentions. Most QA technologies either require large sets of training data to train deep-learning models or require constructions of knowledge graphs. These requirements may present an obstacle to general OAS builders, or at the very least introduce a steep learning curve. Thus, it would be helpful to find a solution that lowers the bar for constructing a QAS.

Finding the right answer to a question in realtime from a large dataset is a challenge. It is a common practice for a QAS to combine information retrieval techniques and keyword matching to identify answer candidates. However, this technique may not return the most appropriate answer in realtime. This motivates us to devise a new method for finding the most appropriate answer in realtime to a user question from a set of candidates.

We present RQAS, a rapid QAS platform over datasets in the form of question-answer pairs (QAP). Based on keyword rankings, RQAS does not involve training, which makes it simple for general QAS builders to construct a QAS over specific domains (a.k.a. vertical QAS). To build a QAS, we simply supply QAP data to RQAS either through an excel file or a web interface. We may also harvest QAPs automatically from community-based QA web sites (Yang and Wang, 2017).

In particular, RQAS first preprocesses QAPs and extracts keywords. It saves the preprocessed data and the original QAPs in a search-engine database. Upon receiving a question, the search engine returns all candidates QAPs that include at least one of the keywords in the user question. We then narrow the range of candidate answers by the rankings of answers. We return the final answer to the question by selecting a QAP with the highest score by exponentially elevating the rankings of certain keywords.

Experiment results show that RQAS achieves a high accuracy in realtime. We also implement a backend system to allow user to easily import QAP data and manage the system.

The rest of the paper is organized as follows: We describe in Section 2 the related work, in Section 3 the architecture of RQAS. We then present in Section 4 the preprocessing procedure, in Section 5 the central processing and in Section 6 our keyword-ranking-elevation algorithm. We report experiment results in Section 7 for evaluations of question-answering accuracy and efficiency. Finally, we conclude the paper in Section 8.

Zhang C. and Wang J.
RQAS: A Rapid QA Scheme with Exponential Elevations of Keyword Rankings.
DOI: 10.5220/0006582203020309
In *Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (KDIR 2017)*, pages 302-309
ISBN: 978-989-758-271-4

2 RELATED WORK

Natural language QAS, first studied in the mid 1960's (Simmons, 1965; Simmons, 1970), is an active research area. For example, Zajac (Zajac, 2001) devised an ontology-based semantic framework for QAS. Moldovan (Moldovan and Novischi, 2002) presented a method for constructing a QAS using WordNet. Rinaldi et al (Rinaldi et al., 2003) described a knowledge-based QAS from a collection of documents. Lopez et al (Lopez et al., 2005) presented an Ontology-Portable QAS for the Semantic Web. Parthasarathy (Parthasarathy and Chen, 2007) devised a QAS based on entity recognitions and thematic analysis. Ferrucci et al (Ferrucci et al., 2010) offered an overview of a knowledge-based QAS for IBM Watson. Recently, Hixon et al (Hixon et al., 2015) presented a QAS based on knowledge graphs. Wang et al (Wang et al., 2010) presented a machine-learning method to model semantic relevance among QAPs. Zhou et al (Zhou et al., 2015) proposed a method to select answers in community question answering systems using convolution neural networks. Mishra (Mishra and Jain, 2016) classified QAS techniques into four categories: information retrieval, knowledge retrieval, data mining, and natural language understanding

Among these techniques, the information-retrieval method is the most studied. Salton et al (Salton et al., 1993) described several passage-retrieving methods in a full-text information system. Tellex et al (Tellex et al., 2003) studied QAS based on information retrievals and discovered that most systems can be divided into the following four components: question analysis, document retrieval, passage retrieval, and answer extraction. These methods use documents as QA data. Michael (Michael Kaisser, 2004) presented approaches using linguistic syntax-based patterns for QAS. Jeon et al (Jeon et al., 2005) presented a method for a translation-based model to retrieve an answer to a question based on similarities of answers to find similar questions in the question and answer archives. Recently, Jovita et al (Jovita et al., 2015) presented a QAS using a vector-space model; however, this system does not meet the realtime requirement, where each retrieval process takes around 10 to 30 seconds of time. Devi and Dua (Devi and Dua, 2016) implemented a QAS using similarity and classification methods, which only works on the Hindi Language. Abdi et al (Abdi et al., 2016) presented a QAS which integrates NLP, ontology and information retrieval technologies, which only applies to the physics domain.

Early methods were focused on the accuracy of retrieving an answer to a user question and seldom paid attention to the time complexity of getting an answer. We would like to construct a QAS that can provide an accurate answer to a user question in realtime. For this purpose, we focus on QAS over any specific domain to help reduce answer-searching time (see Section 5.2 for more discussions).

3 SYSTEM ARCHITECTURE

Assume that we already have sets of QAPs available over different domains. These data sets may be provided by users or harvested from crawling community-based question-answering web sites. It is likely that in some QAPs, the answers do not match well with the corresponding questions. Even if in each QAP, the answer matches well with the question, we note that a question asked by the QAS user may not match well with any existing question in the QAPs. This presents a problem if we simply select an answer to an existing question in a QAP with the highest similarity to the user question, for there may be multiple existing questions with the same highest similarity and the corresponding answers address different things, all of which are less appropriate compared to an answer in another QAP whose question has a smaller similarity to the user question. This is evident from the following example:

Suppose that a user asks the following question: "What is diabetes and symptoms?" In the database about diabetes, the question of each of the following two QAPs has the highest cosine similarity of 0.85 with the user question:

- QAP1:

  Q: What is diabetes diet?

  A: A diabetes diet is a healthy-eating plan that's naturally rich in nutrients and low in fat and calories. Key elements are fruits, vegetables and whole grains.

- QAP2:

  Q: What is diabetes treatment?

  A: Blood sugar monitoring, insulin and oral medications. Eating healthy diet, maintaining a healthy weight and participating in regular activity also are important factors in managing diabetes.

We can see that none of the answers in these two QAPs is appropriate.

On the other hand, the answer in the following QAP, whose question has a smaller cosine similarity of 0.81 with the user question is actually the correct answer:

- QAP3:
  Q: What is the definition of diabetes mellitus?
  A: Diabetes is a group of metabolic disorders in which there are high blood sugar levels over a prolonged period. Symptoms of high blood sugar include frequent urination, increased thirst, and increased hunger.

Taking these two issues into consideration, we should avoid selecting an answer from directly computing similarities of questions.

RQAS consists of a preprocessing component, a central-processing component, and a database shared by both components (see Figure 1). The preprocessing component, shown on the left-hand pane, processes keywords of QAPs, computes a weight for each QAP, and stores all processed data in the database. The central-processing component, shown on the right-hand pane, consists of four modules: question analysis, answer searching, answer scoring, and answer selection. Questions asked by QAS users (also referred to as user question) are forwarded to the question analysis module, which converts the question to a set of keywords. The answer searching module searches for the keywords from the database, and returns all candidate QAPs that contain at least one keyword. The answer scoring module evaluates candidate QAPs and narrows the candidate pool. Finally, the answer selection module selects the most appropriate answer as the final answer.

We use inverted indexing for the database to facilitate realtime searching.

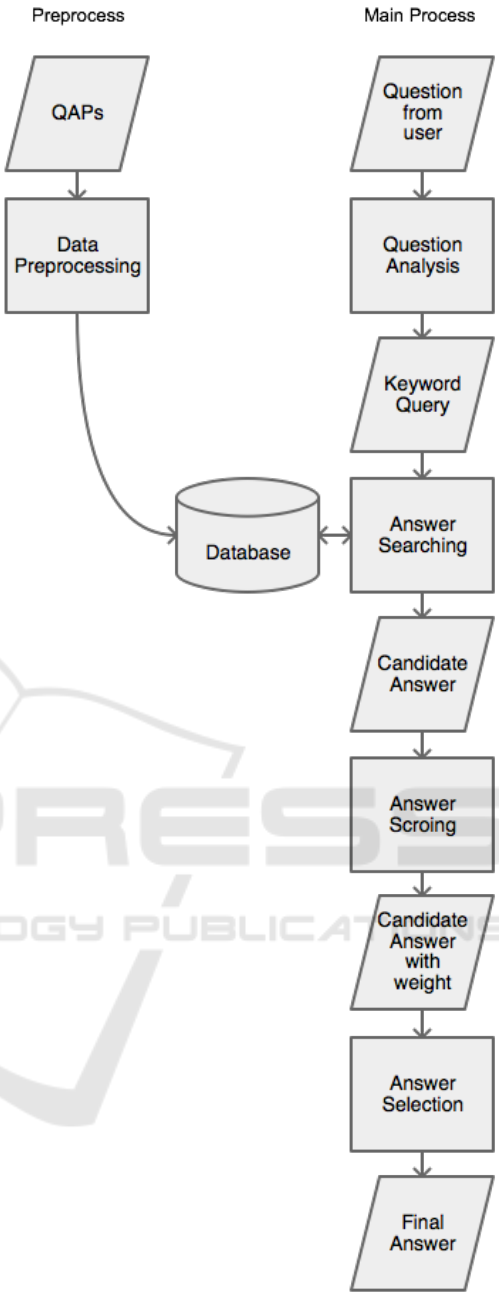

Figure 1: RQAS architecture and data flow.

4 PREPROCESSING

The first step of data preprocessing is to segment phrases (including words) and eliminate stop words (such as "the", "of").

In what follows, we will simply use *keywords* to denote segmented phrases and words extracted by a keyword extraction algorithm, denoted by w with or without subscripts. In general, give a block of text t, we use K_t to denote the multi-set of keywords extracted from t. We use $d = (q,a)$ to denote a QAP with or without subscripts, where q is the question and a is the answer. Note that we allow q to be empty, which means that we allow the system to take in a block of text as an answer. Thus, K_q and K_a denote, respectively, the multi-set of keywords extracted from q and the multi-set of keywords extracted from a. Without loss of generality, we may also use d to denote $K_q \cup K_a$.

Given a QAP $d = (q,a)$, compute the following three values:

1. Term frequency (TF). The TF value of w for each keyword $w \in d$ is computed as follows:

$$\mathrm{TF}(w \in d) = \sqrt{f(w)}, \qquad (1)$$

where $f(w)$ is the frequency of w in $K_q \cup K_a$.

2. Inverse document frequency (IDF). Let N be the total number of QAPs in the dataset. For each keyword $w \in d$, let $F(w)$ denote the number of

QAPs containing w. The IDF value of w is computed as follows:

$$\text{IDF}(w) = 1 + \ln\left(\frac{N}{F(w)+1}\right). \quad (2)$$

3. Document length normalization (DLN). Let d be a QAP and denote by $|d|$ the number of keywords contained in the multi-set $K_q \cup K_a$. Compute DLN of d as follows:

$$\text{DLN}(d) = \frac{1}{\sqrt{|d|}}. \quad (3)$$

The DLN is the inverse square root of the length of the QAP. A longer QAP will result in a smaller DLN. If a keyword appears in a shorter QAP and also a longer QAP, then we consider that the shorter QAP is more important than the longer QAP. The value of DLN reflects this consideration.

For each keyword $w \in d$, compute its TF-IDF value as follows:

$$\text{TF-IDF}(w) = \text{TF}(w) \cdot \text{IDF}^2(w) \cdot \text{DLN}(d).$$

Next, we extract keywords from each QAP and calculate the initial score for each keyword. Through extensive experiments, we find that the RAKE (Rose et al., 2010) and TextRank (Mihalcea and Tarau, 2004) keyword extraction algorithms work well over short passages. Moreover, while RAKE runs slightly faster than TextRank, we find that TextRank works better than RAKE for QAPs written in Chinese. As RQAS is primarily targeted at the English and Chinese languages, we will use TextRank to extract keywords. Examples of keywords extraction using TextRank are shown below, where the numbers are assigned by TextRank and only the top keywords are shown:

- Q: How can we prevent diabetes?

  A: Eat healthy food, do more exercise, and control weight; doing so can help prevent or delay type-2 diabetes for adults at high risk of diabetes.

  Keywords: (diabetes, 0.52), (eat healthy, 0.36), (food, 0.21), (prevent, 0.20)

- Q: What is diabetes?

  A: Diabetes is a group of metabolic disorders with high blood sugar levels over a prolonged period. Symptoms of high blood sugar include frequent urination, increased thirst, and increased hunger.

  Keywords: (high blood sugar levels, 0.24), (symptoms, 0.19), (thirst, 0.15), (hunger, 0.15)

The size of the entire database of QAPs maybe be large, and so preprocessing QAPs can save tremendous computing time for the central-processing component.

Finally, each QAP and the corresponding list of keywords with TextRank and TF-IDF scores are combined into one data item and saved in the database.

5 CENTRAL PROCESSING

After preprocessing QAPs, RQAS waits for user questions.

5.1 Question Analysis

Upon receiving a user question Q, the question analysis module eliminates stop words from Q, applies TextRank on Q to obtain as a multi-set of keywords K_Q, and pass both Q and K_Q to the answer searching module.

5.2 Answer Searching

Upon receiving Q and K_Q from the answer analysis module, the answer searching module performs a searching over the keywords in the database, which returns all QAPs that contain at least one of the keywords.

Let D denote the set of the returned QAPs from the search. We first narrow D. For each QAP $d \in D$, compute its TF-IDF score with respect to Q as follows (Jones, 1972) (Robertson, 2004):

$$\text{TF-IDF}(d|Q) = \sum_{w \in d \cap W_Q} \text{TF-IDF}(w),$$

where the values of TF-IDF(w) were already computed in the preprocessing module and stored in the database. The TF-IDF value of d with respect to Q is referred to as the *searching score* under Q, where Q may be omitted when there is no confusion.

The search engine uses the inverted index (Zobel and Moffat, 2006) data structure and the already computed values of TF-IDF of keywords, and so searching can be done in realtime.

Through extensive experiments, we observe that the most appropriate answer to the user question over a specific domain is typically contained in a very small number of QAPs with top searching scores. In particular, this number is between 10 and 30. To achieve realtime performances, we will only consider the set of top n QAPs for a small fixed value of n, referred to as the *candidate set*. We choose a default value of n to be 20. Depending on the actual dataset of QAPs over a given domain, n may be smaller. If we consider a general dataset of multiple domains, then n may be large, which may affect realtime responds.

We also note that in a QAP, the critical keywords may only appear in the question and not in the answer. Thus, it is possible that a QAP contains the most appropriate answer to the user question, but is excluded from the candidate set. For example, suppose that the user question is "What should I do to prevent diabetes?" and the following QAP is in the dataset:

Q: How do we prevent diabetes?

A: Exercise regularly, eat healthy, ...

In this example, the critical keyword "prevent diabetes" appears in the question but not in the answer, If the answer is long, then the searching score of this QAP may be too small to be selected as a candidate. Thus, if a keyword of the user question also appears in the question component of QAP, then regardless its searching score, add QAP into the candidate set. Let m denote the number of QAPs in the candidate set.

5.3 Answer Scoring

The module receives the candidate set to the user question Q, with a searching score with respect to Q for each candidate QAP and a TextRank score for each keyword in the candidate. We use the following data structure to represent candidate QAPs:

QAP_1	question	
	answer	$keyword_1$: TextRank score
		$keyword_2$: TextRank score
		...
		$keyword_n$: TextRank score
	searching score	
QAP_2	...	
...	...	
QAP_m	...	

Next, we adjust the weight for each candidate QAP, with the searching score as the initial weight. In particular, we elevate the scores of certain keywords so that more important QAPs have larger weights. The reader is referred to Section 6 for details of the algorithm.

After the weight is adjusted for each candidate QAP, append the weight to each candidate QAP, and return the candidate set with new weights to the next module.

5.4 Answer Selection

The answer selection module is responsible for selecting the final answer. This module is needed to handle the situations when more models are introduced to rank QAPs. For the current system setup, we only have one model, so it simply returns the answer component of the QAP with the highest weight as the final answer to the user question Q.

6 ELEVATE KEYWORD RANKINGS

We would want to elevate the rankings of certain keywords, so that the most appropriate QAP for the user question Q will have the highest ranking.

The algorithm involves the following four types of parameters:

1. A list of keywords K_Q from the user question Q.
2. A set of candidate QAPs with respect to Q.
3. A list of keywords extracted from each candidate QAP.
4. A weight of each keyword in each candidate QAP.

We define the following notations.

- Given a QAP $d = (q,a)$, and a keyword $w \in K_q \cup K_a$, let $r(w)$ and $t(w)$ denote, respectively, the TF-IDF value of w and the TextRank score of w. let $s_d(w)$ denote the new score of w with respect to d.
- Let
$$r_d^* = \max\{r(w) \mid w \in d\},$$
$$t_d^* = \max\{t(w) \mid w \in d\}.$$
- Let f_d^* denote the final score of d.

Given a candidate $d = (q,a)$, for each keyword $w \in K_Q$, we have the following three cases:

Case 1: If $w \in K_q$, let

$$s_d(w) = r(w) + 2^{\lambda \cdot t_d^*},$$

where $\lambda \geq 1$ is a fixed constant.

Case 2: If $w \notin K_q$ but $w \in K_a$, then let

$$s_d(w) = r(w) + 2^{t_d^*}.$$

Case 3: If $w \notin K_q$ and $w \notin K_a$, then let

$$s_d(w) = r(w).$$

The constant coefficient λ in Case 1 serves the following purpose: When w appears in K_q, then (q,a) should be elevated to a higher score compared to the case when $w \notin K_q$. To figure out an appropriate value of λ, we tested the value of λ in the range of 1 to 2.5 with a small increment. We found that, through extensive experiments, when the value of λ is less than 2, some answers returned are not satisfactory; when $\lambda = 2$, all answers returned are satisfactory in the experiments. We therefore choose $\lambda = 2$.

Note that in Case 1 we do not further distinguish the case whether $w \in K_a$. The reason is the following: Suppose that $w_1, w_2 \in K_q$. If $w_1 \in K_a$ but $w_2 \notin K_a$, then it is likely that $r(w_1) > r(w_2)$. Hence, it is likely that $s_d(w_1) > s_d(w_2)$, which is what we desire.

Finally, let

$$f_d^* = \sum_{w \in K_Q} s_d(w).$$

Let (q^*, a^*) be a candidate with the largest f^*, then a^* is returned as the answer to user question Q.

7 EVALUATIONS

We use human judgment to evaluate the accuracy and efficiency of RQAS. We collected 138 QAPs in the Chinese language about diabetes as evaluation data.

The criteria of evaluation are listed as follows:

- S: RQAS returns a satisfactory answer.
- F1: RQAS returns an irrelevant answer but it has relevant QAPs.
- F2: RQAS returns an irrelevant answer and it contains no relevant QAPs.

7.1 Experiment 1

To evaluate the accuracy of the answers returned by RAQS, we recruited 10 evaluators of with solid diabetes knowledge. They were asked to glance the answer part of every QAP for the purpose of knowing what knowledge points are contained in the QAPs, without looking at the question part. Then each evaluator asked 10 questions on their own in their own way about the knowledge points they got from the glance, and judged whether the RQAS returned a satisfactory answer. The evaluation results are shown in Table 1.

Table 1: Evaluation results in Experiment 1.

Evaluators	S	F1	F2	Success Ratio
1	9	0	1	90%
2	10	0	0	100%
3	9	1	0	90%
4	7	2	1	70%
5	9	1	0	90%
6	9	1	0	90%
7	6	1	3	60%
8	10	0	0	100%
9	6	0	4	60%
10	10	0	0	100%

The results indicate that different evaluators saw different success ratios. This may be caused by people having different memories. Thus, we devised another experiment.

7.2 Experiment 2

We asked a volunteer to browse the QAPs in the diabetes domain and extract keywords of knowledge points, such as “blood glucose”, “diets”, “protect eyes”. We then provided these keywords to the evaluators, based on which they asked any question in their own way. Examples of the questions asked by the evaluators include “How to regulate blood sugar level?”, “What is blood sugar?”, and “How can I control blood sugar level?”, to name just a few. Results are shown in Table 2.

Table 2: Evaluation results in Experiment 2.

Evaluators	S	F1	F2	Success Ratio
1	8	1	1	80%
2	7	2	1	70%
3	8	1	1	80%
4	8	2	0	80%
5	7	2	1	70%
6	8	0	2	80%
7	7	1	2	70%
8	7	2	1	70%
9	6	2	2	60%
10	7	1	2	70%

The results show that the average success ratio is 73%. We also note that 13% of the failure results are F2. This indicates that our dataset does not have sufficient number of QAPs. If the dataset of QAPs covers every knowledge points in the diabetes domain, the success ratio will be substantially higher.

7.3 Experiment 3

Next, we evaluate the runtime for RQAS to return an answer. We carried out this experiment on a desktop computer with an Intel Core I5 2.6Ghz CPU and 8Gb memory. We tested the runtime to answer one question over datasets with 100 QAPs, 1,000 QAPs, 5,000 QAPs, and 10,000 QAPs. For each dataset we asked 10 different questions and the results are shown idddd n Table 3, where “Min (sec)” means the minimum runtime in seconds, “Max (sec)” the maximum runtime in seconds, and ‘Avg (sec)” the average runtime in seconds.

The results show that the runtime is about the same regardless the size of the dataset, with slightly more time on larger dataset. This indicates the real-time robustness of RQAS.

Table 3: Runtime evaluation results in Experiment 3.

Dataset	Min (sec)	Max (sec)	Avg (sec)
100	0.012	0.031	0.021
1,000	0.016	0.027	0.022
5,000	0.017	0.025	0.022
10,000	0.018	0.026	0.023

8 CONCLUSIONS

We presented RQAS, a rapid QA scheme based on exponential elevations of keyword rankings. Our experiments show that RQAS is efficient and achieves high accuracies. We may improve accuracy and efficiency of RQAS on the following four aspects:

1. Explore other methods to compute QAP rankings for a given question and improve the best QAP by aggregating multiple models. For example, instead of treating keywords as basic units, we may compute QAP rankings by treating sentences as basic units.
2. Optimize data structure.
3. Add semantic analysis to enhance accuracy.
4. Explore question classification models to improve efficiency.

ACKNOWLEDGEMENTS

This work was supported in part by Eola Solutions Inc. The authors are grateful to members of the Text Automation Lab at UMass Lowell for discussions.

REFERENCES

Abdi, A., Idris, N., and Ahmad, Z. (2016). QAPD: an ontology-based question answering system in the physics domain. *Soft Computing*.

Devi, R. and Dua, M. (2016). Performance evaluation of different similarity functions and classification methods using web based hindi language question answering system. *Procedia Computer Science*, 92:520–525.

Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J., Schlaefer, N., and Welty, C. (2010). Building watson: An overview of the DeepQA project. *AI Magazine*, 31(3):59.

Hixon, B., Clark, P., and Hajishirzi, H. (2015). Learning knowledge graphs for question answering through conversational dialog. pages 851–861.

Jeon, J., Croft, W. B., and Lee, J. H. (2005). Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 84–90, New York, NY, USA. ACM.

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1).

Jovita, Linda, Hartawan, A., and Suhartono, D. (2015). Using vector space model in question answering system. *Procedia Computer Science*, 59:305–311.

Lopez, V., Pasin, M., and Motta, E. (2005). *AquaLog: An Ontology-Portable Question Answering System for the Semantic Web*, pages 546–562. Springer Berlin Heidelberg, Berlin, Heidelberg.

Michael Kaisser, T. B. (2004). Question answering by searching large corpora with linguistic methods.

Mihalcea, R. and Tarau, P. (2004). TextRank: Bringing order into texts. In *Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing*.

Mishra, A. and Jain, S. K. (2016). A survey on question answering systems with classification. *Journal of King Saud University - Computer and Information Sciences*, 28(3):345 – 361.

Moldovan, D. and Novischi, A. (2002). Lexical chains for question answering. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Parthasarathy, S. and Chen, J. (2007). A web-based question answering system for effective e-learning. In *Seventh IEEE International Conference on Advanced Learning Technologies (ICALT 2007)*. IEEE.

Rinaldi, F., Dowdall, J., Hess, M., Mollá, D., Schwitter, R., and Kaljurand, K. (2003). Knowledge-based question answering. In *Lecture Notes in Computer Science*, pages 785–792. Springer Berlin Heidelberg.

Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for idf. *Journal of Documentation*, 60:503–520.

Rose, S., Engel, D., Cramer, N., and Cowley, W. (2010). Automatic keyword extraction from individual documents. In Berry, M. W. and Kogan, J., editors, *Text Mining. Applications and Theory*, pages 1–20. John Wiley and Sons, Ltd.

Salton, G., Allan, J., and Buckley, C. (1993). Approaches to passage retrieval in full text information systems. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '93, pages 49–58, New York, NY, USA. ACM.

Simmons, R. F. (1965). Answering english questions by computer: A survey. *Commun. ACM*, 8(1):53–70.

Simmons, R. F. (1970). Natural language question-answering systems: 1969. *Communications of the ACM*, 13(1):15C30.

Tellex, S., Katz, B., Lin, J., Fernandes, A., and Marton, G. (2003). Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 41–47, New York, NY, USA. ACM.

Wang, B., Wang, X., Sun, C., Liu, B., and Sun, L. (2010). Modeling semantic relevance for question-answer pairs in web social communities. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1230–1238, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yang, W. and Wang, J. (2017). Generating appropriate question-answer pairs for chatbots harvested from community-based qa sites. In *Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K), 2017 9th International Joint Conference on*.

Zajac, R. (2001). Towards ontological question answering. *Proceedings of the workshop on ARABIC language processing status and prospects -*.

Zhou, X., Hu, B., Chen, Q., Tang, B., and Wang, X. (2015). Answer sequence learning with neural networks for answer selection in community question answering. *CoRR*, abs/1506.06490.

Zobel, J. and Moffat, A. (2006). Inverted files for text search engines. *ACM Comput. Surv.*, 38(2).