Learning Good Decompositions of Complex Questions

Yllias Chali, Sadid A. Hasan, and Kaisar Imam

University of Lethbridge
Lethbridge, AB, Canada
{chali,hasan,imam}@cs.uleth.ca

Abstract. This paper proposes a supervised approach for automatically learning good decompositions of complex questions. The training data generation phase mainly builds on three steps to produce a list of simple questions corresponding to a complex question: i) the extraction of the most important sentences from a given set of relevant documents (which contains the answer to the complex question), ii) the simplification of the extracted sentences, and iii) their transformation into questions containing candidate answer terms. Such questions, considered as candidate decompositions, are manually annotated (as good or bad candidates) and used to train a Support Vector Machine (SVM) classifier. Experiments on the DUC data sets prove the effectiveness of our approach.

1 Introduction

Complex questions address an issue that relates to multiple entities, events and complex relations between them while asking about events, biographies, definitions, descriptions, reasons etc. [1]. However, it is not always understandable, to which direction one should move to search for the complete answer. For example, a complex question like *"Describe the tsunami disaster in Japan."* has a wider focus without a single or well-defined information need. To narrow down the focus, this question can be decomposed into a series of simple questions such as *"How many people had died by the tsunami?"*, *"How many people became homeless?"*, *"Which cities were mostly damaged?"* etc. Decomposing a complex question automatically into simpler questions in this manner such that each of them can be answered individually by using the state-of-the-art Question Answering (QA) systems, and then combining the individual answers to form a single answer to the original complex question has been proved effective to deal with the complex question answering problem [2,3]. However, issues like judging the significance of the decomposed questions remained beyond the scope of all the researches done so far.

It is understandable that enhancing the quality of the decomposed questions can certainly provide more accurate answers to the complex questions. So, it is important to judge the quality of the decomposed questions and then, investigate more methods to enhance their quality. In this research, we address this challenging task and come up with a supervised model to automatically learn

G. Bouma et al. (Eds.): NLDB 2012, LNCS 7337, pp. 104–115, 2012.

good decompositions of complex questions. For training data generation, at first, we determine the most important sentences from a given set of relevant documents (that contains the answer to the considered complex question), and then, simplify these sentences in the second step. In the third step, questions are generated from the simplified sentences. These questions, considered as candidate decompositions of the complex question, are manually annotated (as good or bad candidates) and used to train a SVM classifier. Experiments on the DUC (Document Understanding Conference[1]) data sets prove the effectiveness of our approach. The remainder of the paper is organized as follows: Section 2 presents a brief survey of related works. Section 3 illustrates the overview of our approach. Section 4 describes the training data generation phase. Section 5 discusses the supervised model used to accomplish the considered task. Section 6 presents the evaluation results and finally, we conclude the paper in Section 7.

2 Related Work

Question decomposition has been proved effective to deal with the complex question answering problem in many studies. For example, in [2], they introduce a new paradigm for processing complex questions that relies on a combination of (a) question decompositions; (b) factoid QA techniques; and (c) Multi-Document Summarization (MDS) techniques. Necessity and positive impact of question decomposition have been also shown in many studies in the complex question answering domain [4,3]. However, there is no direct research on the problem of automatically learning good decompositions of complex questions rather several paraphrasing and textual entailment methods can be considered the most relevant tasks that have been researched. For example, in QA systems for document collections a question could be phrased differently than in a document that contains its answer [5]. Studies have shown that the system performance can be improved significantly by taking such variations into account [6]. Techniques to measure similarities between texts can be an effective way of judging the importance of a sentence. Semantic roles typically offer a significant first step towards deeper text understanding [7]. There are approaches in "Recognizing Textual Entailment", "Sentence Alignment" and "Question Answering" that use semantic information in order to measure the similarity between two textual units [8]. This indeed motivates us to find semantic similarity between a document sentence and a complex question while selecting the most important sentences. Simplifying a sentence can lead to more accurate question generation [9], so we simplify the complex sentences in this research. Different methods have been proposed so far to accomplish the task of Question Generation (QG) [10,11]. Some QG models utilize question generation as an intermediate step in the question answering process [12]. Generated questions can be ranked using different approaches such as statistical ranking methods, dependency parsing, identifying the presence of pronouns and named entities, and topic scoring [9,13].

[1] http://duc.nist.gov/

3 Overview of Our Approach

The main contribution of this research is to develop a classification system that given a complex question and a list of simple questions can decide whether questions in the latter are to be considered a decomposition of the former, viz. they ask about part of the information asked by the complex question. To accomplish this task, simple questions are generated from the documents containing possible answers to the complex question and used to train the classifier[2]. There is a pipeline starting with a complex question and ending with a set of (presumably) one or more simple questions. We assume that a set of relevant documents is given along with each complex question that certainly possesses potential answers to the complex question. However, it is still necessary to identify the most important sentences due to the presence of a huge number of sentences in the data set. We conduct a shallow and a deep semantic analysis between the complex question and the given document sentences to perform this task. Sentences that are found during this process can be long and complex to deal with. Hence, we pass the selected sentences through a sentence simplification module. Once we find the simple sentences, we use a sentence-to-question generation approach in order to generate corresponding questions. We claim that these questions are the potential candidate decompositions of the complex question. We judge the quality of the generated decompositions manually and annotate them into two classes: good candidate and bad candidate, considering their correctness at the question level and verifying whether they can actually satisfy the information need stated in the original complex question partially. We employ a well-known supervised learning technique: SVM, that is trained on this annotated data set and then, the learned model is used to identify good decompositions of the unseen complex questions automatically. Learning good decompositions of complex questions is unique and to the best of our knowledge, no other study has investigated this challenge before in our setting.

4 Training Data Generation

4.1 Filtering Important Sentences

We use a shallow and a deep semantic analysis to extract the most important sentences related to the complex question from the given document collection. We parse the document sentences (and the question) semantically using the Semantic Role Labeling (SRL) system, ASSERT[3]. We use the Shallow Semantic Tree Kernel (SSTK) [7] to get a similarity score between a document sentence and the complex question based on their underlying semantic structures. To do a deeper semantic analysis of the text, after getting stemmed words by using

[2] The same procedure is applied to generate the test data set.
[3] Available at `http://cemantix.org/assert`

the *OAK* system [14] for each sentence and the complex question, we perform keyword expansion using WordNet[4] [15]. For example, the word "happen" being a keyword in the question "What happened?" returns the words: *occur, pass, fall out, come about, take place* from WordNet. Thus, we find out the similar words between the sentence-question pair that gives us a similarity score. We combine this score with the score obtained from the shallow semantic analysis and select the top-scored sentences from the documents. For example[5], *"With economic opportunities on reservations lagging behind those available in big cities, and with the unemployment rate among Native Americans at three times the national average, thousands of poor, often unskilled Native Americans are rushing off their reservations."* is selected as an important sentence.

4.2 Simplifying the Sentences

Sentences that we select in the earlier stage may have complex grammatical structure with multiple embedded clauses. Therefore, we simplify the complex sentences with the intention to generate more accurate questions. We call the generated simple sentences, *elementary sentences* since they are the individual constituents that combinedly possess the overall meaning of the complex sentence. We use the simplified factual statement extractor model[6] [9]. Their model extracts the simpler forms of the complex source sentence by altering lexical items, syntactic structure, and semantics and by removing phrase types such as leading conjunctions, sentence-level modifying phrases, and appositives. For example, for the complex sentence that was selected as important in Section 4.1, we get one of the possible elementary sentences as, *"Thousands of poor, often unskilled Native Americans are rushing off their reservations".*

4.3 Sentence-to-Question Generation

Once we get the elementary sentences, our next task is to produce a set of possible questions from them. We claim these questions to be the candidate decompositions of the original complex question. In this research, we work on generating six simple types of questions: *who, what, where, when, whom* and *how much*. We use the OAK system [14] to produce Part-Of-Speech (POS) tagged and Named Entity (NE) tagged sentences. We use these information to classify the sentences by using a sequence of two simple classifiers. The first classifies the sentences into fine classes (Fine Classifier) and the second into coarse classes (Coarse Classifier). This is a similar but opposite approach to the one described in [16]. We define the five coarse classes as: 1) Human, 2) Entity, 3) Location, 4) Time, and 5) Count. Based on the coarse classification, we consider the

[4] `http://wordnet.princeton.edu/`

[5] We look into a running example through out this paper considering the complex question: "Discuss conditions on American Indian reservations or among Native American communities."

[6] Available at `http://www.ark.cs.cmu.edu/mheilman/`

relationship between the words in the sentence. For example, if the sentence has the structure "Human Verb Human", it will be classified as "whom and who" question types. We define a set of ninety basic word-to-word interaction rules to check the coarse classes. We use the POS information to decompose the main verb and perform necessary subject-auxiliary inversion and finally, insert the question word (with a question mark at the end) to generate suitable questions from the given elementary sentence. For example, for the elementary sentence generated in Section 4.2, we get a simple question as: *"Who are rushing off their reservations?".*

5 Supervised Model

For supervised learning techniques, annotated or labeled data is required as a precondition. We manually annotate the generated simple questions (i.e. candidate decomposed questions) into two classes[7]: good candidate and bad candidate, employ the Support Vector Machines (SVM) that is trained on this annotated data set and then, use the learned model to predict good decompositions from the unlabeled candidate set of decompositions (i.e. test data set) automatically. We describe our feature space, learning and testing modules in the following subsections.

5.1 Feature Space

For our SVM classifier, we use a total of thirteen features that are divided into two major categories: one that considers the correctness at the question level and other is a coverage component that measures whether a decomposed question can actually satisfy the information need stated in the original complex question partially. We automatically extract these features from the questions (for both training and testing data) in order to feed them to the supervised models for learning and then, for prediction. These features are related to the original important sentence (that is selected in Section 4.1), the input sentence (elementary sentence), the generated simple question, and the original complex question. Correctness of the questions can be measured using a composition of the following features:

Grammaticality: We count the number of proper nouns, pronouns, adjectives, adverbs, conjunctions, numbers, noun phrases, prepositional phrases, and subordinate clauses in the syntactic structures of the question and the input sentence. We set a certain threshold[8] to denote the limit up to which a candidate can

[7] We inspect each question to measure whether they are lexically, syntactically and semantically correct or not. We also judge each decomposed question against the original complex question and analyze further to find out whether it can ask for any information that can be found in the given data. This analysis guides us to label each question as $+1$ (good candidate) or -1 (bad candidate).

[8] The thresholds are set after inspecting the questions and the input sentences manually.

be termed as good. We also include some boolean features to encode the tense information of the main verb.

Length: We calculate the number of tokens in the question, the original source sentence, the input elementary sentence, and the answer term (that is replaced by a question type). We set a threshold on this value, too.

Presence of Question Word: We consider some boolean features to identify the presence or absence of a certain question type: *who, what, where, when, whom* and *how much*.

Presence of Pronouns: If a question has one or more pronouns, we understand that the question is asking about something that has limited reference and hence, we consider the question as *vague*. To identify whether a question includes pronouns or not, we employ a boolean feature.

The coverage component of our feature extraction module tells whether a decomposed question can satisfy the requested information need partially. To automatically encode this feature for each question, we conduct an extensive linguistic analysis and a deep semantic analysis between the decomposed question and the original complex question.

Linguistic Analysis: We use ROUGE (Recall-Oriented Understudy for Gisting Evaluation) to automatically determine the quality of a question by comparing it to the original complex question using a collection of measures [17].

Deep Semantic Analysis: We conduct a deep semantic analysis between the decomposed question and the original complex question (according to the procedure discussed in Section 4.1) that outputs a similarity score as the feature value.

5.2 Learning and Testing

Once we get the feature values for all the decomposed questions along with the associated annotation (good or bad candidate), we feed this data to the supervised learner so that a learned model is established. We use a set of 523 questions for the training purpose. Later, this model is used to predict the labels for the new set of simple questions automatically during the testing phase. Our test data set includes 350 questions. In this work, we use SVM [18] as the classifier. To allow some flexibility in separating the classes, SVM models have a cost parameter, C, that controls the trade off between allowing training errors and forcing rigid margins. We use the SVM^{light} [19] package[9] for training and testing in this work. SVM^{light} consists of a learning module and a classification module. The learning module takes an input file containing the feature values with corresponding labels and produces a model file. The classification module is used to apply the learned model to new samples. We use $g(x)$, the normalized distance from the hyperplane to each sample point, x to rank the questions.

[9] `http://svmlight.joachims.org/`

6 Evaluation and Analysis

6.1 Corpus

The recent 2005, 2006, and 2007 Document Understanding Conference (DUC) series, run by the National Institute of Standards and Technology (NIST), have modeled a real-world complex question answering task, in which a question cannot be answered by simply stating a name, date, quantity, etc. They provide several topics, a complex question related to each topic, and a set of 25 relevant documents (that contains the answer to the complex question). The task is to synthesize a fluent, well-organized 250-word summary of the documents that answers the question(s) in the topic statement. We use a subset of 10 topics[10] from the DUC-2006 data (that came from the AQUAINT corpus, comprising newswire articles from the Associated Press and New York Times (1998-2000) and Xinhua News Agency (1996-2000)) to run our experiments.

6.2 Cross-Validation

Cross-validation is an effective technique for estimating the performance of a predictive model. We apply a 3-fold cross-validation on the annotated questions (i.e. training data set) to estimate how accurately our SVM model would perform on the unlabeled set of candidate decompositions (test data set). We use a randomized local-grid search [20] for estimating the value of the trade-off parameter C. We try the value of C in 2^i following heuristics, where $i \in \{-5, -4, \cdots, 4, 5\}$ and set C as the best performed value of 0.0625 for the linear kernel[11]. In Figure 1, we show the effect of C on the testing accuracy of the SVM classifier. We can see that the accuracy is largely dependent on the value of the parameter C. The accuracy rises with the increase of the C value and reaches the peak when $C = 0.0625$. However, after that, an opposite trend is visible with the increase of the C value.

6.3 Intrinsic Evaluation

Using the learned model, the supervised SVM classifier automatically predicts the labels (good or bad candidate) of the new set of decomposed questions that are generated for each new complex question. To evaluate the performance of our approach, we manually assess the quality of these questions. Two university graduate students judged the questions for linguistic quality and overall responsiveness following a similar setting to the DUC-2007 evaluation guidelines[12]. The given score is an integer between 1 (very poor) and 5 (very good) and is guided by consideration of the following factors: 1. Grammaticality, 2. Correct question type, 3. Referential clarity (Presence of pronoun), and 4. Meaningfulness.

[10] We use 6 topics for training and 4 topics for testing.
[11] We found linear kernel as the best performer among all kernels.
[12] `http://www-nlpir.nist.gov/projects/duc/duc2007/quality-questions.txt`

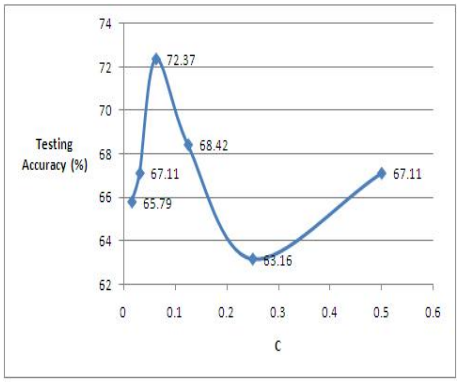

Fig. 1. Effect of C on SVM's testing accuracy

They also assigned a content responsiveness score to each question. This score is also an integer between 1 (very poor) and 5 (very good) and is based on the factor whether the question helps to satisfy the information need expressed in the original complex question. This task was performed intuitively by investigating the original complex question, the corresponding simple question at hand, and the given document collection. We compare the top-ranked good questions with the performance of a set of randomly picked (good/bad mixed) questions. For each topic, we also judge the performance of the bad questions alone[13]. Table 1 shows the evaluation results for SVM. Analyzing Table 1, we see that SVM predicted *Good Questions* improve the linguistic quality and responsiveness scores over *Mixed (Random) Questions* by 74.06%, and 100.00%, respectively whereas they outperform the Linguistic Quality and Responsiveness scores over *Bad Questions* by 20.58%, and 12.50%, respectively. These results suggest that the SVM classifier performed well to rank the decomposed questions accurately. We also see that *Bad Questions* outperform the *Mixed (Random) Questions* in terms of linguistic quality because they were small in length and had good grammatical structure. However, they could not beat the responsiveness scores meaning that small questions have limited coverage over the requested information need.

Table 1. Linguistic quality and responsiveness scores (average) for SVM

Categories	Linguistic Quality	Responsiveness
Good Questions	4.10	3.60
Mixed (Random)	2.35	1.80
Bad Questions	3.40	1.60

[13] *Good questions* refer to the top 50% questions with high scores predicted by the SVM classifier, whereas *bad questions* refer to the bottom 50% questions.

In another evaluation setting, the two annotators judge the questions for their overall acceptability as a good or a bad candidate decomposition and assign a score (an integer between 1 (very poor) and 5 (very good)) to each of them. The outcome of this evaluation scale is then converted into a binary (+1/-1) rating scale, deciding that a score above 3 is positive. Then, the accuracy of the SVM's binary prediction is computed with respect to the manual annotation using the following formula:

$$Accuracy = \frac{number\,of\,Correctly\,Classified\,Questions}{Total\,number\,of\,Test\,Questions} \quad (1)$$

We experimented with a total of 226 questions[14] for this evaluation and found 159 of them to be correctly classified by the SVM classifier showing an accuracy of 70.35%. An inter-annotator agreement of Cohen's $\kappa = 0.2835$ [21] was computed that denotes a fair agreement [22] between the raters.

6.4 Extrinsic Evaluation

To determine the quality of the decomposed questions based on some other task such as summarization can be another effective means of evaluation. We call it as extrinsic evaluation. We pass the top-ranked decomposed questions to the Indri search engine (i.e. a component of the Lemur toolkit[15]). For all decomposed questions, Indri returns ranked sentences from the given data set that are used to generate a 250-word summary according to the DUC guidelines. We evaluate these summaries against four human-generated "reference summaries" (given in DUC-2006) using ROUGE [17] which has been widely adopted by DUC. We consider the widely used evaluation measures Precision (P), Recall (R) and F-measure for our evaluation task. Table 2 shows the ROUGE-2 and ROUGE-SU scores of our proposed SVM system as they are used as the official ROUGE metrics in the recent DUC evaluations.

Table 2. ROUGE measures for SVM

Measures	Recall	Precision	F-score
ROUGE-2	0.0675	0.0570	0.0618
ROUGE-SU	0.1027	0.0734	0.0856

Statistical Significance: To show a meaningful comparison, in Table 3, we present the average ROUGE-2 scores (Recall) of our SVM system and the two state-of-the-art systems, *Trimmer* and *HMM Hedge* that use a Multi-Candidate Reduction (MCR) framework for multi-document summarization [23]. These well-known systems do not perform question decomposition, so, we treat them as

[14] We consider only the questions for which both annotators agreed on rating as positive or negative.

[15] Available at `http://www.lemurproject.org/`

the *non-decomposed baselines* to perform system comparisons. An approximate result to identify which differences in the competing systems' scores are significant can be achieved by comparing the 95% confidence intervals for each mean. So, we include the 95% confidence intervals to report the statistical significance of our system. ROUGE uses a randomized method named bootstrap resampling to compute the confidence intervals. Table 3 yields that the SVM system outperforms the two baseline systems. We can also see that the confidence intervals of all the systems overlap with each other meaning the fact that there is no significant difference between our system and the baseline systems.

Table 3. Comparison of different systems

Systems	ROUGE-2
Trimmer	0.0671 [0.06332–0.07111]
HMM Hedge	0.0625 [0.05873–0.06620]
SVM	0.0675 [0.06548–0.07221]

6.5 Feature Engineering

To analyze the impact of different features, we run another experiment considering the SVM classifier generated top-ranked questions. In Figure 2, we plot a graph to show the performance of different systems considering several variants of the feature space during experiments. In the figure, **Grammaticality**, **Length**, **Pronoun**, and **Coverage** indicate that the corresponding feature is not considered during experiments, whereas **OnlyGrammaticality**, **OnlyLength**, **OnlyPronoun**, and **OnlyCoverage** denote the presence of that particular feature only. **All** denotes the inclusion of all features. From the figure, we understand that if we exclude the *Grammatically* feature, the responsiveness score improves quite a lot, whereas exclusion of the *Length* feature produces good scores for both linguistic quality and responsiveness. On the other hand, if we do not consider the *Pronoun* feature, the scores have a negative impact. Again, omitting the *Coverage* feature decreases the responsiveness score. On the other hand, if we consider *OnlyGrammaticality* feature, we have better linguistic quality and worse responsiveness score. *OnlyLength* feature yields good scores for both linguistic quality and responsiveness scores whereas *OnlyPronoun* feature provides bad scores for both denoting its lower impact on the SVM learner. *OnlyCoverage* feature shows a higher responsiveness score with a moderate linguistic quality score and finally, considering *All* features yields a good linguistic quality while showing a decent performance in terms of responsiveness score. From this comparison of feature combinations, we can conclude that the inclusion of the *Pronoun*, and *Coverage* features helps to achieve the best performance for the considered task. This comparison also suggests that the chosen features are appropriate by themselves, but in combination are not able to produce a qualitative jump in performance.

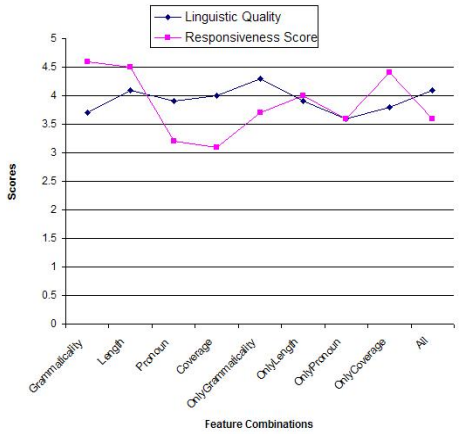

Fig. 2. Graph for different feature combinations

7 Conclusion and Future Works

We propose a supervised model for learning good decompositions of complex questions. We perform a rigorous evaluation and analysis to show the effectiveness of our approach. Furthermore, we analyze the impact of different features on the performance of the SVM classifier and conclude that the combination of the *Pronoun*, and *Coverage* features yields the best performance. In future, we plan to use more sophisticated features while investigating the potentials of other supervised approaches such as Conditional Random Fields (CRF), and Hidden Markov Models (HMM) for the learning task.

Acknowledgments. The research reported in this paper was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada-discovery grant and the University of Lethbridge.

References

1. Chali, Y., Joty, S.R., Hasan, S.A.: Complex Question Answering: Unsupervised Learning Approaches and Experiments. Journal of Artificial Intelligence Research 35, 1–47 (2009)
2. Harabagiu, S., Lacatusu, F., Hickl, A.: Answering complex questions with random walk models. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 220–227. ACM (2006)
3. Hickl, A., Wang, P., Lehmann, J., Harabagiu, S.: Ferret: Interactive question-answering for real-world environments. In: Proceedings of the COLING/ACL on Interactive Presentation Sessions, pp. 25–28 (2006)
4. Lacatusu, F., Hickl, A., Harabagiu, S.: Impact of question decomposition on the quality of answer summaries. In: Proceedings of the Fifth International Conference on Language Resources and Evaluation, LREC 2006 (2006)

5. Pasca, M.: Open-domain question answering from large text collections, 2nd edn. Center for the Study of Language and Information (2003)
6. Harabagiu, S., Hickl, A.: Methods for using textual entailment in open-domain question answering. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics, ACL, pp. 905–912 (2006)
7. Moschitti, A., Quarteroni, S., Basili, R., Manandhar, S.: Exploiting Syntactic and Shallow Semantic Kernels for Question/Answer Classificaion. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pp. 776–783. ACL, Prague (2007)
8. MacCartney, B., Grenager, T., de Marneffe, M., Cer, D., Manning, C.D.: Learning to Recognize Features of Valid Textual Entailments. In: Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, New York, USA, vol. 4148 (2006)
9. Heilman, M., Smith, N.A.: Extracting simplified statements for factual question generation. In: Proceedings of the Third Workshop on Question Generation (2010)
10. Wang, W., Hao, T., Liu, W.: Automatic Question Generation for Learning Evaluation in Medicine. In: Leung, H., Li, F., Lau, R., Li, Q. (eds.) ICWL 2007. LNCS, vol. 4823, pp. 242–251. Springer, Heidelberg (2008)
11. Chen, W., Aist, G., Mostow, J.: Generating Questions Automatically from Informational Text. In: Proceedings of the 2nd Workshop on Question Generation (AIED 2009), pp. 17–24 (2009)
12. Hickl, A., Lehmann, J., Williams, J., Harabagiu, A.: Experiments with interac- tive question-answering. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics ACL 2005, pp. 60–69 (2005)
13. McConnell, C.C., Mannem, P., Prasad, R., Joshi, A.: A New Approach to Ranking Over-Generated Questions. In: Proceedings of the AAAI Fall Symposium on Question Generation (2011)
14. Sekine, S.: Proteus Project OAK System (English Sentence Analyzer) (2002), http://nlp.nyu.edu/oak
15. Fellbaum, C.: WordNet - An Electronic Lexical Database. MIT Press, Cambridge (1998)
16. Li, X., Roth, D.: Learning Question Classifiers. In: Proceedings of the 19th International Conference on Computational Linguistics, Taipei, Taiwan, pp. 1–7 (2002)
17. Lin, C.Y.: ROUGE: A Package for Automatic Evaluation of Summaries. In: Proceedings of Workshop on Text Summarization Branches Out, Post-Conference Workshop of Association for Computational Linguistics, Barcelona, Spain, pp. 74–81 (2004)
18. Cortes, C., Vapnik, V.N.: Support Vector Networks. Machine Learning 20, 273–297 (1995)
19. Joachims, T.: Making large-Scale SVM Learning Practical. In: Advances in Kernel Methods - Support Vector Learning (1999)
20. Hsu, C., Chang, C., Lin, C.: A Practical Guide to Support Vector Classification, National Taiwan University, Taipei 106, Taiwan (2008), http://www.csie.ntu.edu.tw/cjlin
21. Cohen, J.: A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement 20(1), 37–46 (1960)
22. Landis, J.R., Koch, G.G.: The Measurement of Observer Agreement for Categorical Data. Biometrics 33(1), 159–174 (1977)
23. Zajic, D., Dorr, B.J., Lin, J., Schwartz, R.: Multi-candidate reduction: Sentence compression as a tool for document summarization tasks. Information Processing and Management 43, 1549–1570 (2007)