

Using CBR for Portuguese Question Generation

Daniel Diéguez^{1,2}, Ricardo Rodrigues¹, and Paulo Gomes¹

¹ CISUC, University of Coimbra, Portugal,

² Higher School of Computer Engineering, University of Vigo, Spain,
danielsda@gmail.com, {rmanuel,pgomes}@dei.uc.pt

Abstract. In this paper, we propose a new architecture for Question Generation for the Portuguese Language. This architecture aims at the automatic generation of questions, to be used later, for instance, in automatic question answering by means of predictive question generation. Our approach combines a case-based reasoning system and a module for question generation. The question generation module uses manually built rules that are fed to the case-based reasoning engine for selecting which ones should be used. This is accomplished by comparing the answer and the sentence part-of-speech tag sequences. An identical tag sequence on sentences and answers usually implies a similar sequence on the corresponding questions. We discuss the details of this architecture, how it performs and the results obtained so far.

Keywords: automatic question generation, case-based reasoning, natural language processing.

1 Introduction

In recent years, Question Generation (QG) [14, 12] has received significant attention by researchers in the field of Natural Language Processing (NLP) [8], becoming an important research area with potential for use in advanced learning technologies, such as intelligent tutoring systems [17], dialogue systems [5], and educational technologies [4].

QG aims to analysing text, identifying sentences and topics that are then used to formulate questions regarding those same sentences and their topics. QG also allows to validate knowledge. For instance, Wolfe [17] presents a “computer aid to independent study”, which applies QG for tutoring systems. In this system, questions are generated to be used later in the evaluation of students knowledge about a given topic.

In this work, we use a Case-Based Reasoning (CBR) [1] system for generating questions, using similarity between part-of-speech (POS) tag sequences of answers to FAQ³ questions and sentences in a corpus, assuming there are similarities also between the questions on FAQs and those to be automatically generated.

³ We use the term FAQ (Frequently Asked Questions) in a broader sense, representing question-answer pairs.

In this paper, we start by introducing background knowledge about QG and CBR. In section 3, we present how to carry out the construction of CBR system for QG. Section 4 dicusses the results obtained evaluation. Section 5 provides related work. And finally, in Section 6 we present some concluding remarks.

2 Background Knowledge

This section presents a summary of the fundamental concepts and the technology employed in our work. The two main topics we will discuss are: Question Generation and Case-based Reasoning.

2.1 Question Generation

In the field of QG [14, 12], there are several approaches for carrying out the task of generating questions. In many cases, the nature of automatic question generation depends on the task within which it is embedded. Automatic QG deals with the analysis of text, identifying sentences and topics, that are then used to formulate questions regarding those same sentences and their topics. Nowadays, automatic question generation is used in multiple situations, such as pedagogical environments, intelligent tutoring systems, conversation with virtual agents and information retrieval.

Over the last years this topic has seen renewed interest. However it remains mainly in the domain of advanced learning technologies, such as intelligent tutoring systems, inquiry-based environments, and game-based learning environments, extending only what had been done before. These systems continue to be used essentially to evaluate students [15, 3].

2.2 Case-Based Reasoning

CBR [1] is a problem-solving paradigm that uses specific knowledge from previous experiences, i.e., situations of a particular problem (cases). When a new problem arrives it is compared to the existing base in search of similar cases. If found, the existing cases are reused for the current situation. Adicional benefit of such situation is that after solving a problem the CBR systems add it to their knowledge base thus extending their capabilities in context of future queries.

CBR has been formalized for purposes of computer reasoning as a four-step process [1]:

1. **Retrieve:** this step retrieves the most similar case or cases which are contained in knowledge base. A case consists of a problem, its solution, and, typically, annotations about how the solution was derived.
2. **Reuse:** this step reuses the information and knowledge (obtained in the previous step) in that case to solve the problem.
3. **Revise:** this step comprises the task of revising the proposed solution by the previous step, making the necessary changes for correcting mistakes in the solution from the previous step.

4. **Retain:** this step retains the useful experience for future reuse, and the case base is updated by a new learned case, or by modification of some existing cases.

3 Question Generation Using CBR System

For the task of QG, we propose the architecture shown in Fig. 1. We can see in it the four modules that compose our system. The **Annotation Module** annotates the corpus written in natural language. After that, this module generates documents with the annotated corpus. The **CBR Module** is composed by the CBR system, which uses a knowledge base for solving the cases resulting from the first module. A case in our CBR system comprises an answer and a question. The goal of the CBR system is to generate a question for each case. For this, the CBR system uses the **Question Generation Module**, which tries to generate a question through an answer. This module is based in rules. Finally, the **Pattern search Module** discovers patterns in the annotated corpus that facilitate the task of building rules.

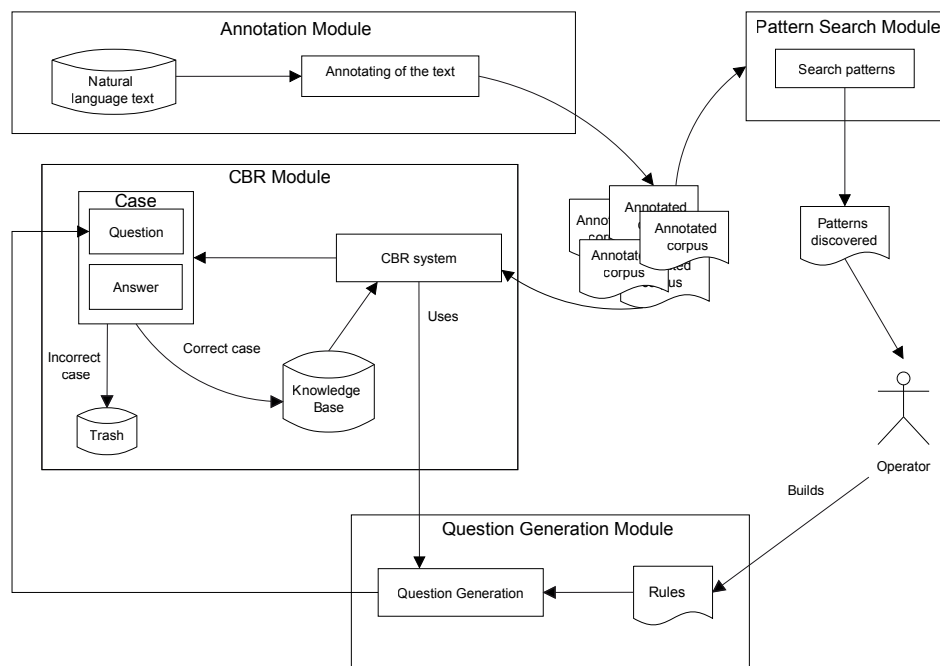


Fig. 1. System Global Architecture.

3.1 Annotation Module

For annotating the corpus, we use multiple NLP tools. These tools are a POS tagger, a lemmatizer and a sentence splitter. Our goal is to use these tools for doing a depth analysis of a whole corpus. This provides the possibility of creating data structures containing knowledge about the different levels of analysis existing in the whole corpus, such as morphological analysis, syntactic analysis and semantic analysis. Once the text is separated in sentences and annotated, we deploy a sentence parser for identifying the subject, the main verb, the predicate and the predicate complements. The sentence parser also determines the voice of the sentences, which will later help on the process of generating questions. Identifying the sentence voice is an important fact because it is not the same to generate a question for sentences on active voice than for sentences on passive voice.

To understand better the annotation process, we present an example of a sentence extracted from a text of the corpus that we used. The sentence is “Diogo está a ter aulas de dicção para ficar com pronúncia americana” (Diogo is attending to a pronunciation class for practising the american accent) and the annotated sentence is shown in Table 1.

Table 1. Example of an annotated sentence by Annotation Module.

Sentence	Diogo está a ter aulas de dicção para ficar com pronúncia americana.						
Subject	Diogo						
Main verb	está a ter						
Predicate	está a ter aulas de dicção para ficar com pronúncia americana						
Voice	Active voice						
Words	Diogo	está	a	ter	...	americana	.
Tags	prop	v-fin	prp	v-inf	...	adj	punc
Lemmas	Diogo	estar	a	ter	...	americano	.

Once these tasks finish, all the information obtained through this process is stored in files for feeding the next modules. We store this information in files because the time spent annotating the corpus is large and most of the time, it only has to be done once – in fact, we use the output of this module as input to the CBR Module and to the Pattern Search Module.

3.2 CBR Module

In the **first step**, we apply a similarity metric for determining if there are similar cases in the knowledge base to the problem that we want to solve. The **similarity metric** used for determining the similarity is based in the Damerau-Levenshtein distance algorithm[7]. Our goal is to calculate the distance between two sentences (a sentence in the corpus and an answer in the FAQ) at the level of the POS

tags of each word in the answer, for determining the syntactic similarity of both sentences.

Our reasoning is based on the assumption that given a question Q , with an answer A , if a sentence A' has the same POS tag sequence of A , it could probably become an answer to an automatically generated question Q' , using the sentence elements found in A' .

At this point, we have to make a decision regarding *what is the correct distance between two answers at the level of the POS tags?* This is something that can vary, mostly if other metrics are applied to below or simply if other techniques are used in the reuse step of CBR cycle for determining if the case to solve can generate a question. In Section 4, we explain the results obtained applying different distances between answers and we also explain the reason we have chose those distances.

In the **second step** we check if the retrieval process was successful, i.e, if similar cases were recovered from the knowledge base. If so, then we know that the case to solve is a candidate for generating a question. Then this case will be analyzed by the module of QG which will try to build a question. Once that happens, we have two possible results: a generated question or no generated question. If the first situation occurs, this means that we have a solved case and the CBR system should analyze it in the next step (revise step) to be validated. Otherwise, the CBR system discards the current case and tries it with another case until finishing all the cases.

The **third step** is responsible of reviewing the solved cases previously. Currently, we are performing this process manually, because we still do not have the resources for automatic validation. This step allows the CBR system to learn. If the solved case is correct, it is marked for the next step (retain process).

The **fourth step** consists simply in adding to the knowledge base the cases which were reviewed successful in the previous step. When that happens, the CBR system increases the number of cases as examples for resolving new problems. This will allow to the CBR system to identify new cases.

3.3 Question Generation Module

The question generation module is composed of two different parts. The first part focuses on what is really necessary for formulating a question in natural language – in this case, for Portuguese. For instance, the questions “*Para que serve um computador?*” (What is the purpose of a computer?), “*Onde mora o Presidente da República Portuguesa?*” (Where does the President of the Portuguese Republic live?) and “*Quem ganhou?*” (Who has won?) can be structured as shown in Fig. 2.

Observing the different questions of the figure and other questions, we consider that a question, at least for the Portuguese Language, can be composed by the following elements:

- **Preposition:** a preposition that precedes the interrogative pronoun.
- **InterrogativePronoun:** any interrogative pronoun.

Question	Quem ganhou ?
Question Structure	interrogative pronoun verb question mark
Question	Para que serve um computador ?
Question Structure	preposition interrogative pronoun verb object question mark
Question	Onde mora o Presidente da República Portuguesa ?
Question Structure	interrogative pronoun verb object question mark

Fig. 2. Examples of questions with its question structure.

- **Verb:** main verb of a eventual answer.
- **Object:** subject, predicate or predicate complements of an answer.
- **?:** question mark.

Regarding the elements that can compose a question, we define the following question structure shown in Fig. 3. This structure defines the preposition and the object as optional elements and the interrogative pronoun, the verb and the question mark as mandatory elements.

[\$preposition] \$interrogativePronoun \$verb [\$object] ?

Fig. 3. Question structure

Examples that use the question structure are shown in Fig. 2, where we can see as the question “*Onde mora o Presidente da República Portuguesa?*” does not have the preposition component, but has the remaining components: interrogative pronoun, verb, object and question mark. On the other hand, the question “*Para que serve um computador?*” has all the elements defined in the question structure.

The second part of the question generation module is composed of the question generation algorithm and the rules that allow the transformation of an eventual answer into a question, using the question structure previously defined. The process for getting a question consists of applying the algorithm until finding a rule that transforms an eventual answer into a question. On the other hand, if there is not a rule that transforms an eventual answer into a question, the question generation module sends a failure to the reuse step of the CBR system, indicating that the case to be solved does not have a solution.

Before we describe the algorithm, we must define what a rule is, and how it is built. A rule comprises a **search pattern** and a **operation set**. A **search pattern** is a combination of word(s), tag(s) or lemma(s), i.e., a different combinations of the elements that exist in an annotated answer. For instance, “*ser adj*” (“*ser*” is the lemma of the verb to be and “*adj*” is a tag that means adjective) is a combination of a lemma and a tag. An operation set is composed by several actions that allow the transformation of an answer into a question. To understand better how rules work, we can observe in Fig. 4 how a rule with the search pattern “*começar em*” (“*começar*” is the lemma of the verb to start and “*em*” is a preposition that means in) transforms the sentence “*A Segunda Guerra Mundial começou em 1939*” (The Second World War began in 1939) into a question applying the operation set.

Answer	A Segunda Guerra Mundial começou em 1939
Search pattern	começar em
Operation set	Apply NER about predicate NER finds a date (1939) preposition = (empty) interrogative pronoun = Quando verb = começou object = a Segunda Guerra Mundial
Question structure	\$preposition \$interrogativePronoun \$verb \$object?
Question	Quando começou a Segunda Guerra Mundial?

Fig. 4. Example of transformation of an answer into a question.

In the operation set, we use Named Entity Recognition (NER) [11] on the subject or predicate (depending on how the rule is built) for determining the interrogative pronoun of the question. In this example, we apply NER on the predicate. As NER recognizes a date, the interrogative pronoun is substituted by “*Quando*” (when). After that, we replace the verb by “*começou*” and the object by the subject of the sentence in this case “*A Segunda Guerra Mundial*”. Once applied the operation set of the rule, we obtain the question “*Quando começou a Segunda Guerra Mundial?*” (When did the Second World War begin?). The rules are an important part of the module of question generation, but a major problem arises in the construction of the rules. The problem is that it is hard to identify the search patterns. To resolve this problem, we created the **Pattern Search module**, which is described in Section 3.4.

After introducing what a rule is, we can explain how the question generation algorithm works. The algorithm aims to discover search patterns in a structure composed of words, tags and lemmas. Its structure is basically an array of annotated words. An annotated word is an object with a word, a tag and a lemma. For instance, the word “*foi*” (was) corresponds to the following annotated word: word(*foi*), tag(*v-fin*), lemma(*ser*).

The strategy of the questions generation algorithm is the following. For all input cases of the module of question generation, it is built an array with all the annotated words that an answer contains. For each search pattern, we check if they exist in the array. If the search pattern is found, the next step is to apply the operation set of the rule to which belongs the search pattern, for transforming the answer into a question. Otherwise, if the search pattern is not found, the question generation module sends a failure to the reuse step of the CBR system.

In algorithm 1 is shown the pseudocode of the algorithm.

Algorithm 1 Algorithm for Automatic Question Generation

Require:

Eventual Answer *answer*
 Rule set $R = \{r_1, \dots, r_n\}$
 Annotated Word set $W = \{p_1, \dots, p_n\}$ from *answer*

Ensure: *question*

```

sp =  $\emptyset$  (a pattern search)
os =  $\emptyset$  (a operation set)
result =  $\emptyset$ 
question =  $\emptyset$ 
for all  $r_i$  in  $R$  do
  sp  $\leftarrow$  search pattern from  $r_i$ 
  result  $\leftarrow$  search(sp,  $W$ )
  if result true then
    os  $\leftarrow$  operation set from  $r_i$ 
    question  $\leftarrow$  apply(os, answer)
    return question
  end if
end for
return question

```

An example of the search function from the algorithm 1 is shown in Fig. 5. In this example, the search function tries to discover the **search pattern** “*começar em*” in the array of the annotated words. When a **search pattern** is found and then, the operation set is applied for generating the question.

3.4 Pattern Search Module

The rules used by the **Question Generation Module** are built manually, but the **search pattern** of each rule is discovered automatically by this module. This module uses an algorithm for searching patterns in the annotated corpus. The algorithm discovers the number of times that different combinations of words, tags or lemmas are repeated in the whole corpus. For this, we use the annotated words that were defined previously.

The strategy of the algorithm is as follows. Select a sentence from the annotated corpus. For each sentence, the algorithm chooses the possible combinations

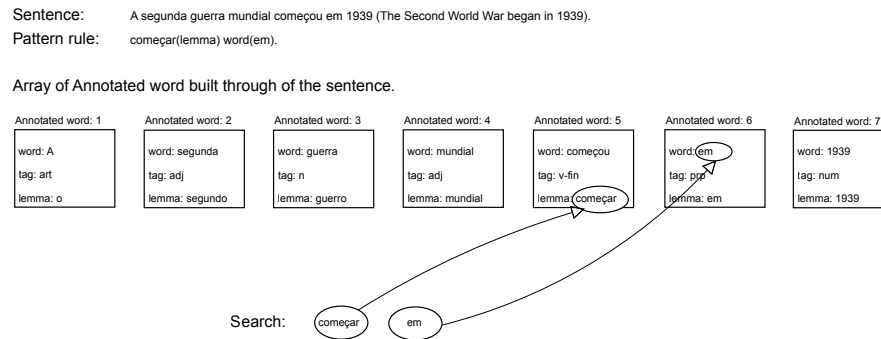


Fig. 5. Example of working question generation algorithm.

from the annotated word corresponding to the verb and its following next annotated words. Once a combination is selected, the algorithm looks for it in the rest of the sentences of the corpus. The goal is to count the number of times that a combination is repeated in all the corpus. When the algorithm finishes, it is made a rank of the combinations discovered, specified by the number of times that a combination is repeated in all the corpus.

The combinations become patterns and the next step is to store them in a file. This file is reviewed manually for us for building the rules that will be used by the question generation algorithm.

An example of a pattern found by the algorithm is “*ser adj*”. We use this pattern for building the rule in Fig. 6.

Search pattern	ser adj
Operation set	preposition = (empty) interrogative pronoun = Como (How) verb = sentence verb object = sentence subject
Question structure	Como \$verb \$object?

Fig. 6. Rule example.

As we can observe, this pattern indicates that all sentences that end with the verb “ser” (to be) followed by an adjective automatically generate the question structure of the Fig. 6. For instance, if we apply the rule of the Fig. 6 to the sentence “*o concerto do Rui Veloso foi espectacular*” (Rui Veloso’s concert was spectacular) we obtain the question “*Como foi o concerto do Rui Veloso?*” (How was the concert of Rui veloso?).

4 Experiments

To perform the experiments, our initial idea was to build a knowledge base using several collections of FAQs extracted automatically from Internet. But when we made the first tests, we observed that the results were not as good as expected. Many of the generated questions were odd or did not make any sense. This happens because FAQs sometimes contain question-answer pairs that are not related correctly. Sometimes, the answers do not begin with the appropriate words or, in other cases, the answers are very long. Usually, this information is not directly related to the question, and therefore our approach generated odd questions. Another important aspect is that FAQs usually do not have all question types (“When”, “Who” and “Where”, questions are unusual in FAQs”). These are some of the reasons why we did not have good results using this procedure. Therefore, we considered the possibility of building a knowledge base manually, with questions and answers written and tagged by us. We also considered that all questions should have a concrete answer, i.e., the answers are entirely related to the questions. An example of a question-answer pair built manually is: “*Quem é o Presidente da República Portuguesa?*” (Who is the President of The Portuguese Republic?) “*O Presidente da República Portuguesa é Cavaco Silva*” (The President of the Portuguese Republic is Cavaco Silva).

With the new approach, two different experiments were done, having different distances in the Levenshtein distance algorithm. In the first experiment, we used a distance of 1 and on the second experiment a distance of 2. The reason we chose these distances and not others is simple. The answers that we have in our knowledge base have an average of 7 tokens by answer. Knowing this, we considered an error margin no bigger than 30% for avoiding over matching. As such, the determined distances were 1 and 2. With distance 1, we assume an average error of an 14% and with the distance 2, we assume an error of an 28%. The corpus used as input to the CBR system is the same for both experiments. The corpus is extracted from Newspaper *Público* available through CHAVE [13]. The knowledge base is manually created, and it contains ten question-answer pairs. The knowledge base is also the same for the two experiments. The results obtained are shown in Table 2.

Table 2. Number of generated cases

Experiment	Number of generated cases
1	686
2	4317

To validate the cases, we used a sample with a confidence level of the 95% and a confidence interval of the 5%. Data obtained are showed in Table. 3. The samples were distributed to five persons. The criterion used for validating the cases is as follows:

Table 3. Data of the sample sizes determined

	Experiment 1	Experiment 2
Confidence level	95%	95%
Confidence interval	5%	5%
Population	686	4317
Sample size	246	353

- **Incorrect case:** an incorrect case is a case that contains an incorrect question and incorrect answer. An incorrect question is one that has no meaning at the semantic level or its construction is incorrect at the syntactic level.
- **Correct question:** cases where the question is correct but the pair is not totally correct, i.e., the answer does not answer correctly to the question. A correct question is one that is built correctly both at the semantic level and the syntactic level.
- **Correct case:** a correct case is a case that contains a correct question and a correct answer.

After validating the cases, the results are presented in Table 4.

Table 4. Results obtained

	Experiment 1	Experiment 2
Incorrect cases	16%	27%
Correct question	25%	19%
Correct cases	59%	54%
Total correct generated questions	84%	73%

As we can see, the results for experiment 1 are better in all aspects than experiment 2. This is because the distance used on the experiment 1 is lower than the experiment 2. Therefore, we observe that the shorter the distance, better results are obtained. But an important fact is that the number of generated cases in the experiment 2 is greater than the experiment 1. This is because in this case, the distance used is greater and therefore more case are examined. In experiment 1, the CBR system generated 686 cases, while experiment 2 generated 4317 cases.

5 Related Work

Regarding related work, Aliet al. [2] describe a QG system where given a sentence, the system generates a set of questions for which the sentence contains, implies, or needs answers. For this, they build elementary sentences from the input complex sentences using a syntactic parser. Later, to encode necessary information of each sentence, they use NER, a POS tagger and classify the sentences based on their subject, verb, object and preposition for determining the

possible type of questions to be generated. Kalady et al. [9] in their work present an approach to question generation based on syntactic and keyword modeling and describe how to generate different types of question from a single input sentence. In Heilman and Smith [6] it is presented an extensible approach to generating questions for the purpose of reading comprehension assessment and practice.

Other researchers such as Wang et al. [16] generate the questions automatically based on question templates which are created by training on many medical articles. Silveira et al. [14] describe a general framework for characterizing and situating efforts to automatically generate question from free text. That work proposes to generate questions regardless of the domain and language used. However, other researchers such as Nielsen and Buckingham [10] define a question taxonomy based on a priori educational research and the analysis of tutoring transcripts from multiples domains which is totally dependent of the domain in that case, tutoring education systems. Other works also follow the lines of Nielsen and Buckingham [10] in terms of domain dependence. In [3], an interesting approach is described to automatically generate questions for vocabulary assessment.

6 Conclusions

In this paper, we propose a new architecture for automatic QG. Our system uses a CBR system for determining which sentences can be used by a module of question generation. The module of QG transforms the sentences into questions. After that, we obtain cases with question-answer pairs. The architecture here proposed generates question without defining any specific domain. Other important aspect is that this architecture can be used with other languages, at least, languages that follow a *subject, verb and object* structure. For this, only the file of rules should be changed. Also, This system serves for creating a corpus of question-answer pairs for the Portuguese Language through its knowledge base which increases with each run.

Regarding the results obtained, we consider that these results are a good start for this approach, because we get to generate many correct questions, concretely a 84% in the experiment 1 and 73% in the experiment 2.

One challenge for this new architecture proposed for automatic QG will be feeding to a Automatic System Question Answering, through the knowledge base, with the aims of improving the results already obtained by this kind of systems. The current system can also be improved by adding more and better rules, we only are using a small set of rules. So one of the goals is to incorporate new rules that permit generate more questions.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Commun* pp. 39–59 (1994)

2. Ali, H., Chali, Y., , Hasan, S.A.: Automatic question generation from sentences: a preliminary approach. In: Conference on Traitement Automatique de la Langue Naturelle. Montreal, Canada (19-23 July 2010)
3. Brown, J.C., Frishkoff, G.A., Eskenazi, M.: Automatic question generation for vocabulary assessment. In: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing. pp. 819–826. HLT '05, Association for Computational Linguistics, Stroudsburg, PA, USA (2005), <http://dx.doi.org/10.3115/1220575.1220678>
4. Graesser, A.C., Chipman, P., Haynes, B.C., Olney, A.: AutoTutor: an intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions on Education* 48(4), 612–618 (2005), <http://dx.doi.org/10.1109/TE.2005.856149>
5. Graesser, A.C., VanLehn, K., Rosé, C.P., Jordan, P.W., Harter, D.: Intelligent tutoring systems with conversational dialogue. *AI Mag.* 22, 39–51 (October 2001), <http://portal.acm.org/citation.cfm?id=567363.567366>
6. Heilman, M., Smith., N.A.: Question Generation via Overgenerating Transformations and Ranking. Tech. rep., Language Technologies Institute, Carnegie Mellon University (2009)
7. Hyyrö, H.: A bit-vector algorithm for computing levenshtein and damerau edit distances. *Nordic J. of Computing* 10, 29–39 (March 2003), <http://portal.acm.org/citation.cfm?id=846090.846095>
8. Jurafsky, D., Martin, J.H.: *Speech and Language Processing*. Pearson Education International, Inc., Upper Saddle River, New Jersey, USA, second edn. (2008)
9. Kalady, S., Elikkottil, A., Das, R.: Natural language question generation using syntax and keywords. In: Proceedings of QG2010: The Third Workshop on Question Generation. Kerala, India (2010)
10. Nielsen, R.D., Buckingham, J., Knoll, G., Marsh, B., Palen, L.: A Taxonomy of Questions for Question Generation. In: Rus, V., Graesser, A. (eds.) Proceedings of the Workshop on the Question Generation Shared Task and Evaluation Challenge. Arlington, Virginia, USA (September 2008)
11. Poibeau, T., Kosseim, L.: Proper Name Extraction from Non-Journalistic Texts. In: Daelemans, W., Sima'an, K., Veenstra, J., Zavrel, J. (eds.) Proceedings of the 11th Computational Linguistics in the Netherlands Meeting (CLIN 2001). pp. 144–157. Editions Rodopi B.V., Amsterdam, New York, USA (2001)
12. Rus, V., Graesser, A.C. (eds.): *The Question Generation Shared Task and Evaluation Challenge*. The University of Memphis (2009)
13. Santos, D., Rocha, P.: CHAVE: topics and questions on the Portuguese participation in CLEF. In: Peters, C., Borri, F. (eds.) Cross Language Evaluation Forum: Working Notes for the CLEF 2004 Workshop (CLEF 2004). pp. 639–648. IST-CNR, Pisa, Italy (15-17 September 2004), <http://www.linguatca.pt/documentos/SantosRochaCLEF2004WN.pdf>, revised as Santos & Rocha (2005)
14. Silveira, N.: Towards a Framework for Question Generation. In: Rus, V., Graesser, A. (eds.) Proceedings of the Workshop on the Question Generation Shared Task and Evaluation Challenge. Arlington, Virginia, USA (September 2008)
15. Stanescu, L., Spahiu, C.S., Ion, A., Spahiu, A.: Question Generation for Learning Evaluation. In: Proceedings of the International Multiconference on Computer Science and Information Technology. pp. 509–513. IEEE Press, Wisa, Poland (October 2008)
16. Wang, W., Hao, T., Liu, W.: Automatic question generation for learning evaluation in medicine. In: Leung, H., Li, F., Lau, R., Li, Q. (eds.) *Advances in Web Based*

- Learning ICWL 2007. Lecture Notes in Computer Science, vol. 4823, pp. 242–251. Springer Berlin Heidelberg (2008)
17. Wolfe, J.H.: An Aid to Independent Study through Automatic Question Generation (AUTOQUEST). Tech. rep., Navy Personnel Research and Development Center, San Diego, California, USA (October 1975)