Bootstrapping Multiple-Choice Tests with THE-MENTOR

Ana Cristina Mendes, Sérgio Curto and Luísa Coheur

Spoken Language Systems Laboratory - L^2F/INESC-ID
Instituto Superior Técnico, Technical University of Lisbon
R. Alves Redol, 9 - 2º– 1000-029 Lisboa, Portugal
{ana.mendes,sergio.curto,luisa.coheur}@l2f.inesc-id.pt

Abstract. It is very likely that, at least once in their lifetime, everyone has answered a multiple-choice test. Multiple-choice tests are considered an effective technique for knowledge assessment, requiring a short response time and with the possibility of covering a broad set of topics. Nevertheless, when it comes to their creation, it can be a time-consuming and labour-intensive task. Here, the generation of multiple-choice tests aided by computer can reduce these drawbacks: to the human assessor is attributed the final task of approving or rejecting the generated test items, depending on their quality.
In this paper we present THE-MENTOR, a system that employs a fully automatic approach to generate multiple-choice tests. In a first offline step, a set of lexico-syntactic patterns are bootstrapped by using several question/answer seed pairs and leveraging the redundancy of the Web. Afterwards, in an online step, the patterns are used to select sentences in a text document from which answers can be extracted and the respective questions built. In the end, several filters are applied to discard low quality items and distractors are named entities that comply with the question category, extracted from the same text.

1 Introduction

Multiple-choice tests are an effective technique for knowledge assessment, requiring a short response time and with the possibility of covering a broad set of topics. Typically, these tests consist in a number of test items, each composed by two parts: a question and a group of suggested answers. Respondents are supposed to identify the correct answer among the incorrect ones (called distractors). The following is an example of a multiple-choice test item with one correct answer and two distractors:

Q. *"What is the largest ocean?"*

1. *Atlantic* `(distractor)`
2. *Pacific* `(correct answer)`
3. *Indian* `(distractor)`

The manual creation of multiple-choice test items is a time consuming trial and error process; in this context, computer aided multiple-choice tests generation can help reducing the amount of time allocated to this task.

In this paper, we hypothesize that the process of generating multiple-choice tests with only one correct answer per question can rely on the bootstrap of a set of question/answer (Q/A) seed pairs. Here, we describe our approach to automatically generate multiple-choice tests and present THE-MENTOR, a system that generates multi-choice tests about a free text document. THE-MENTOR is composed by two main components which perform the following tasks:

Learning lexico-syntactic patterns – A set of Q/A seeds is used to bootstrap patterns that relate questions with answers. Patterns are learned from the Web, and we exploit its redundancy to create plausible patterns. Moreover, we perform verb expansion and allow several types of patterns, according to the precision of the match against the original seeds. The decision of accepting different types of patterns resulted from the fact that, if patterns are too specific (`strong` patterns), they will not frequently match and not many tests will be generated; if patterns are too generic (`weak` patterns), the quality of the generated tests decreases.

Generation of test items – The retrieved patterns extract sentences where answers can be found and from which the respective questions can be built. In order to discard low quality items, several filters are applied. Distractors are named entities that comply with the question category, extracted from the same text.

Afterwards, the user can evaluate the quality of the generated test items through a web interface. The multiple-choice test will be composed by the test items the user considered as having quality.

This paper is organized as follows: in Section 2 we describe the pattern learning task; in Section 3 we describe how to generate multiple-choice test items. In Section 4 we show the evaluation results; in Section 5 we present related work; in Section 6 we present our conclusions and point to future work directions.

2 Pattern Learning

The first task in the generation of multiple-choice test items by THE-MENTOR has to do with learning lexico-syntactic patterns and is performed off-line, that is, before the user specifies the text document based on which the test will be generated.

The algorithm for pattern learning is based on the bootstrapping technique presented in [1], and involves the following two stages. First, we use a seed pair – composed by a natural language question and its correct answer – to bootstrap patterns that relate questions and answers. We call B-PATTERNS to the patterns extracted from the bootstrap process. Second, the B-PATTERNS are validated using a validation pair – also a natural language question and its correct answer – as input. The validation pair will help to remove those patterns that are too specific to the seed pair. Therefore, each seed pair has a validation pair associated. These pairs are automatically grouped, knowing that although their constituents are lexically distinct, they must share the same syntactic structure as well as the same category. For example, the seed question *"Who is the President of France?"* cannot be validated by the question *"What is the capital of France?"* since, although both share the same syntactic structure (*WHNP VBZ NP*), they have a different focus: the former searches for the name of an individual (HUMAN:INDIVIDUAL) and the later for the name of a city (LOCATION:CITY).

The syntactic analysis of questions is made using the Berkeley Parser [2] trained on the QuestionBank [3], a treebank of 4,000 parse-annotated questions. In what concerns the classification of questions, we use a machine learning-based classifier fed with features derived from a rule-based classifier [4]. Regarding the question categories, we use Li and Roth's two-layer taxonomy [5], consisting of a set of six coarse-grained categories and fifty fine-grained ones.

Finding patterns The algorithm to find B-PATTERNS starts by generating permutations of a set comprising the seed answer, the phrasal nodes of the seed question (excluding the *Wh*-phrase), and a wildcard * which stands as a placeholder for one or more words and adds diversity into the generated patterns. For instance, considering the question *"Who painted the Birth of Venus ?"* and the sentence *Botticelli has painted the Birth of Venus*, a wildcard is required to match the verb *has*. Since we do not allow the wildcard to be the first or the last element in the query, the total number of permutations is $n! - 2(n-1)!$, in which n is the number of elements to be permuted. In addition, the reason why we use phrasal nodes instead of question tokens as it is done in [1], is because they represent a single unit of meaning, and therefore should not be broken down into parts (except for verb phrases). For example, considering the previous question *"Who painted the Birth of Venus ?"*. it does not make sense to divide the noun-phrase *the Birth of Venus*, since it would generate several meaningless permutations, like *Birth the painted Botticelli * of Venus*.

After the permutations have been created, each is enclosed in double quotes and sent to Google search[1]. The double quotes ensure that each search result contains the exact quoted permutation, with every word in the exact same order in which it appears in the original query. The snippets retrieved from the search engine are then broken down into sentences, and if there exists a sentence that matches the respective permutation, we rewrite it as a pattern. Consider again the question "[$_{\text{WHNP}}$ Who] [$_{\text{VBD}}$ painted] [$_{\text{NP}}$ the Birth of Venus]", and suppose a sentence *Botticelli has painted the Birth of Venus* that matches the permutation *Botticelli * painted the Birth of Venus*. The resulting pattern would then be "{*ANSWER*} has *VBD NP*", created by replacing each phrasal node with the respective tag, and the seed answer with the tag "{ANSWER}".

Pattern Validation While many of the learned patterns are generic enough to be applied to other questions, there are others specific to the seed pair. For instance, the pattern "*NP* was *VBD* around 1486 by {*ANSWER*}", extracted from the sentence *The Birth of Venus was painted around 1486 by Botticelli*. Since this pattern only works for the seed question (and possibly for a small number of works of art of that same year), it should be filtered-out. To eliminate these elements, we use a different algorithm which requires the use of a validation pair.

The validation algorithm works by testing each generated pattern against the validation pair, and calculating its precision. The precision is considered to be the ratio

[1] In this work we use Google as the search engine. However, there is no technical reason that prohibits this system to use another search engine.

between the number of times the pattern matched the retrieved snippets and the number of times the pattern was expected to match (that is, the maximum number of snippets retrieved by the search engine). For example, using the aforementioned pattern and the validation pair *"Who painted Guernica?"/Picasso*, the query *Guernica was painted around 1486 by Picasso* would be issued, resulting in zero results – and thus zero precision. This would cause the pattern to be ruled-out, as the algorithm dictates that each pattern must have precision larger than a threshold in order to be retained.

2.1 Handling Verbs

Our method for learning patterns implies that every phrase (except the *Wh*-phrase) must be present in the B-PATTERNS. From now on, we call `strong` patterns to the B-PATTERNS that contain all phrases (and their contents) of the seed question. Whereas this is the expected behaviour for Noun- and Prepositional-phrases that should be stated *ipsis verbis* in the sentence fragments that will generate the patterns as they are in the seed question, the same does not apply for Verb-phrases. The pattern generator should be flexible enough to capture a pattern in the sentence *Botticelli finished painting The Birth of Venus in 1485.*, even if the surface word that corresponds to the verb is not the same as the one present in *"Who painted the Birth of Venus?"*.

Being so, we allow the verbs in the pattern to be in a different inflexion than the main verb in the seed question. Moreover, in case the seed question has an auxiliary verb, the sentence fragment does not need to contain it, since these are most probable to appear in interrogative sentences than on declaratives ones. To create these patterns, we pick the main verb of the question and conjugate it in its multiple inflexions. Afterwards, a new query is sent to the search engine with the several inflexions, and without the presence of the auxiliary verb (if it exists in the question).

The B-PATTERNS generated by verb inflection are named `inflected` patterns.

2.2 Allowing Weak Patterns

There are, however, some patterns that should not be disregarded, even if they do not contain all question phrases and cannot be handled by allowing the multiple inflexion of verbs. These patterns arise from sentence that, despite not completely rephrasing the question, capture the existing relation between it and the answer. For instance, the pattern "*NP*, by $\{ANSWER\}$" should be recovered from the sentence *The Birth of Venus, by Botticelli*, even if it does not include the verb (in this case, *painted*). These patterns are different from the `strong` and `inflected` patterns, not only because of how they were created, but also because they will trigger distinct strategies in the test item generation. To generate this type of patterns (called `weak` patterns), the procedure is similar as referred, just we do not allow the Verb Phrases in the question to be present on the permutations.

Table 1 shows a set of patterns[2] generated from, and validated with, the snippets retrieved by the search engine, for questions with flatten syntactic structure *WHNP VBD NP* and category HUMAN:INDIVIDUAL.

[2] Here, as well as throughout the entire paper, the Penn Treebank II Tags [6] are used.

Table 1. Example of extracted patterns with respective precision and type.

Question	HUMAN:INDIVIDUAL-*WHNP VBD NP*	
Precision	**Pattern**	**Type**
0.625	{*ANSWER*}'s *NP*	`W`
0.25	{*ANSWER*} began *VBG* NP	`I`
0.625	*NP VBD* by {*ANSWER*}	`S`

3 Building Mutiple-Choice Tests

The next task in the generation of multiple-choice test items by THE-MENTOR is done online. The user specifies the target documents and the system parses the text and applies the learned patterns in order to obtain Q/A pairs (as well as distractors). This method also involves several strategies for filtering the obtained Q/A pairs in order to discard low quality pairs.

3.1 Extracting Question/Answer Pairs

Our algorithm for extracting Q/A pairs relies on matching lexico-syntactic information from the B-PATTERNS against the parsed sentences of the target document. Each match is done at two levels in the sentence parse tree: at the word level, since most of the patterns include tokens to separate the syntactic components, and at the syntactic level. For that purpose, we have developed a tree matching algorithm (out of the scope of this paper) to find all the occurrences of a given pattern on the syntactic tree of the parsed sentences.

After the extraction of the sentence fragments where questions and their respective answers are stated, we apply a set of filters to refine the proceeding generation of multiple-choice test items, namely:

Forcing Question/Answer Category Matching The extracted fragments in which the answer does not comply with the expected category can be discarded. Consider, for example, the Q/A pair: *"Who was François Rabelais?"-An important 16th century writer*. Here, the answer agrees with the semantic class expected by the question (HUMAN:INDIVIDUAL), indicated by the word *writer*.
Thus, we test the answer in order to check if at least one of its words belongs to the question category. By using WordNet's lexical hierarchy, a word is associated with a higher-level semantic concept, which represents itself a question category. To do so, we have manually grouped a set of WordNet synsets into fifty clusters, each representing a question category. For example the category HUMAN:INDIVIDUAL is related with the synsets `person`, `individual`, `someone`, `somebody` and `mortal`. The words *actor*, *leader* and *writer* are hyponyms of (at least) one of these synsets. Since WordNet can be seen as a directed acyclic graph, with synsets as vertices and lexical relations – such as hypernym – as edges, we employ a breadth-first search on the translated synset's hypernym tree, in order to find a synset that pertains to any of the pre-defined clusters.

Discarding Anaphoric References A group of simple regular expressions is used to invalidate questions that contain anaphoric references and others, which we empirically know that will not result in quality multiple-choice test items. Thus, questions like *"What is it?"*, *"Where is there?"* or *"What is one?"* are discarded.

3.2 Generation of Test Items

Given that we successfully discover and extract fragments in the target document that match B-PATTERNS, the generation of multiple-choice test items is straightforward and performed according to the type of the fragment extracted from the sentence (`strong`, `inflected` and `weak`). Since we keep track of both the set of questions (that share the syntactic form) for which we discovered the patterns, and the sentences that generated them, the generation goes as follows: both `strong` and `inflected` patterns result in a direct unification of all extracted fragment components with the B-PATTERNS components. However, within `inflected` patterns, the verb is inflected with the tense and person existing in the question and the auxiliary in the question is also used. In what concerns the `weak` patterns, we perform the unification of the fragment components with the respective pattern components, and for all the components that do not appear in the fragment, the components in the question are used.

Regarding the generation of distractors, we search in the text for named entities whose type agree with the category of the question. For that purpose, and to take advantage of the rich taxonomy of question categories utilized, we developed and use several strategies to recognize named entities from texts. These strategies include the usage of regular expressions (to extract numerical entities), gazetteers (to extract locations) and a machine learning-based recognizer (for persons, locations, organizations and others). We choose the named entities nearer to the sentence that originated the Q/A pair, however not in the same sentence. As an example, if we consider the sentence *This resource briefly explores the telegraph invented by Samuel Morse.*, that originates the Q/A pair *"Who invented the telegraph?"-Samuel Morse*, since its category is HUMAN:INDIVIDUAL, we will search and use as distractors the named entities of type PERSON in the nearer sentences.

4 Experiments

Our approach takes as input natural language questions and their correct answers. In our experiments, we used 139 natural language Q/A pairs, some taken from an on-line trivia, others manually created. All questions are factoids pertaining to 10 categories – ENTITY:CURRENCY (5 Q/A pairs), ENTITY:SPORT (3), ENTITY:LANGUAGE (4), HUMAN:INDIVIDUAL (28), LOCATION:CITY (11), LOCATION:COUNTRY (24), LOCATION:MOUNTAIN (2), LOCATION:OTHER (27), LOCATION:STATE (12) and NUMERIC:DATE (23). To allow comparisons with other systems, the used Q/A pairs are available in `http://qa.l2f.inesc-id.pt/wiki/index.php/Resources`

The pairs were automatically grouped (according to their category and syntactic structure), in order to create the seed/validation pairs. This step resulted in a set of 668 seed/validation pairs, which, along with the different syntactic structures and categories,

Table 2. Example of seed-validation pairs.

Group – HUMAN:INDIVIDUAL-`WHNP VBD NP`
S *"Who wrote Odyssey?"/Homer*
V *"Who painted Guernica ?"/Picasso*
Group – LOCATION:COUNTRY-`WHPP VBD NP VBN`
S *"In which country was Bjorn Borg born?"/Sweden*
V *"In which country was the match invented?"/France*

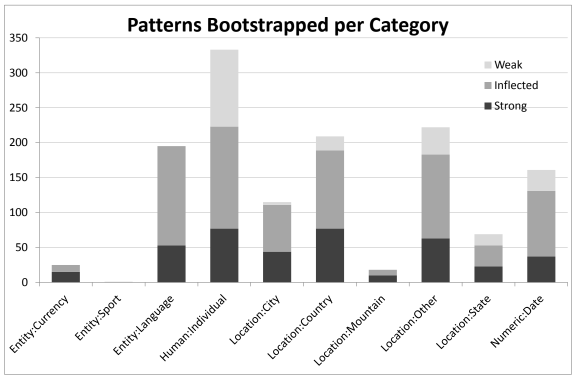

Fig. 1. Distribution of B-PATTERNS per category.

led to a total of 20 groups. Examples of seed/validation pairs are presented in Table 2, with a reference to the group to which they belong.

The 16 top ranked snippets retrieved by the web search engine Google were used to learn the patterns, according to the process described in Section 2. The learning task resulted in 1348 B-PATTERNS, from which 1126 were unique.

The distribution of patterns according to the category is presented in Figure 1. We noticed that the more seed/validation pairs exist, the more B-PATTERNS were bootstrapped by category. An exception to this was the category ENTITY:LANGUAGE, for which a small number of pairs (12) gave rise to a large number of patterns (all belonging to the type `strong` and `inflected`). The highest number of patterns were discovered for the category HUMAN:INDIVIDUAL, which was the category with more seed/validation pairs. Moreover, a great share of patterns of this category are of type `weak`: almost one third. The ratio between `weak` patterns and the total amount of patterns is less for the other categories: in five categories this ratio lower then 5%.

To evaluate the generated test items we used a similar model of test item review to that of [7]. If an item makes no sense (like *"Who left alone much?"-the new British*

Table 3. Number of generated question and answer pairs for each pattern type.

Type	**Degree of Review** *Min.*	*Mod.*	*Maj.*	**Discarded**	**Total**
S	14	26	11	8	**59**
I	2	7	8	28	**45**
W	0	2	55	53	**110**
Total	16	35	74	89	**214**

rulers) it is *discarded*; otherwise, it is marked as *worthy*. A *worthy* item is evaluated according to the degree of review needed, and classified as: **minor**, if it requires up to minimal corrections (like article introduction or spelling corrections), for example *"Who was François Rabelais?"-An important 16th century writer*; **moderate**, if it requires the removal/insertion or reordering of words, or if a set of distractors is not applicable, for example, in: *"Who was Eugène Viollet-le-Duc?"-the associated architect* the words "*the associated*" should be removed; **major**, if it requires a deep grammatical correction, for example in: *"Who became Philip I?"-the Spanish king* the question should be reformulated to *"What did Philip I become?"*.

4.1 Evaluation

We used the Wikipedia article about the "History of Portugal" [3] as target document in the evaluation of our approach for the generation of the test items.

Results for each type of pattern – `strong`, `inflected` and `weak` – are shown in Table 3.

A total of 806 sentences from the referred article tried to match with every of the learned patterns. Considering the type of the involved patterns, 59 were `strong`, 45 `inflected` and 110 `weak` patterns extracted Q/A pairs. Although more patterns were activated, they did not pass the filtering phase.

As expected, most of the Q/A pairs extracted from the `strong` patterns generated items considered as *worthy*. This type of patterns generated the pairs that needed small or no revision. However, a tendency exists for augmenting the degree of review needed when lowering the constraints imposed by the patterns (measured by the existence of the question components in the pattern). The extracted question/answer pairs are distributed through six of the aforementioned categories:

- *Human:Individual*: generated the highest number of pairs (138), mostly using `strong` patterns. Greatly contributed to the total of minor revision items (11 of 14);
- *Location:City*: generated the second highest number of pairs: 27, in which 13 were *worthy*;
- *Location:Country*: generated only eight pairs, only two considered *worthy*;

[3] `http://en.wikipedia.org/wiki/History_of_Portugal`

– *Location:Other*: generated 18 pairs. Since this category is more generic that the previous two *Locations*, it could generate better test items than the others (for instance, a Q/A pair *"Where is Lisbon?"-Europe* is allowed, but not *"In which country is Lisbon?"-Europe*);
– *Location:State*: generated seven pairs. All of them were discarded.
– *Numeric:Date*: generated 13 pairs, three of which needing minor revisions.

Although no patterns of the other four categories matched, this result was somewhat expected: firstly, due to the nature of the document target in use: an article about the history of a country; secondly, the patterns belonging to these categories existed in higher number than the others (an exception being the category ENTITY:LANGUAGE).

The B-PATTERN that generated more patterns was the `weak` pattern "*NP with* {*ANSWER*}", however for the 72 generated, all of them are either discarded, or need major revision. The one that was most successful, with a higher number of *worthy* generated items with minor revision when compared to the total number of generated patterns (11 in 42), was "{*ANSWER*} *VBD NP*", both from the category HUMAN:INDIVIDUAL.

Concerning the distractors, mostly they were appropriate to the generated test item. Moreover, and since they are in agreement with the question, if its *Wh*-phrase has to be reviewed/replaced, the distractor will probably have to be changed too.

4.2 Discussion

The approach used within THE-MENTOR receives as input a set of natural language Q/A pairs and generates test items in order to create multiple-choice tests. With this approach, several test items were generated automatically that can help the creation of multiple-choice tests, originated from a small set of seed pairs. These seeds can be easily found and built, for instance, using the test sets made available in evaluation campaigns for QA systems (like TREC or CLEF).

As results suggested, there is a relation between the types of the B-PATTERNS and the test items they generate: STRONG patterns generated better Q/A pairs, however in a lower number, and WEAK patterns generated Q/A pairs with lower quality, but still most of them can be used after some revision. However, a similar relation could not be spotted for the category type and the generated test items. It was anticipated that the WEAK patterns would lead to the worse results, however we consider that they are able to capture important information. We believe that their posterior generation into Q/A pairs and the automatic filtering phase should be improved.

Also, our approach relies in the lexico-syntactic information stated on the patterns. Even if with this we are neglecting information that could be valuable in the matching of sentences and generation of the tests, for instance semantic information, we could still generate a large set of test items, most of which can be used.

5 Related Work

There are not many examples in the literature of systems that focus on the generation of multiple-choice tests. An exception is the computer-aided environment for generating multiple-choice test items, described in [8]. Authors present a system that relies heavily on natural language processing techniques and resources, built on the notion of key-terms (terms about which the test items should be generated). The system performs three main tasks: it starts by identifying and extracting the key-terms from the source corpora, by using regular expressions that match nouns and noun-phrases; afterwards, question generation rules are applied only to sentences of SV(O) structure and the generated questions filtered to assure grammatical correctness; lastly, concepts semantically related with the answer are retrieved from the WordNet [9]. After the generation, a *post-editing* phase exists in which the test items are revised by human assessors. This system was later adapted to the medical domain [10].

Authors [11] and [12] describe two other systems for multiple-choice test generation. However, the type of questions they output are different from the ones of the aforementioned system and from our work: the fill-in-the-blank (or cloze) questions, are built with blank spaces to be filled by the appropriate option. The first system, called WebExperimenter, obtains distractors from several sources/techniques such as WordNet, edit distance or mutual information, and uses a machine learning classifier to decide the correct position of the blank in the question. WebExperimenter was later adapted to assist the learning of English as a second language [13]. The second system was originally built with the purpose of measuring the English proficiency of non-native speakers, and works by selecting and replacing a word (authors focused uniquely on verbs) in a correct English sentence with a blank. Distractors are chosen in order to maintain the same characteristics of the correct choice, and picked from a thesaurus. The correctness of each distractor is assessed through a web-based verification: if the sentence restored from the blanked sentence and the distractor exists in the web, the distractor is assumed to be correct. Following this line, several systems, like REAP [14] and FAST [15], put their efforts in the improvement of cloze questions.

Although the literature in multiple-choice test generation is not extensive, this task can easily borrow and adapt techniques employed in Question-Answering (QA). These have influenced our work, hence, here we briefly describe some of these systems.

A good parcel of the research in question answering relies on the usage of patterns, namely to bridge the gap between the question and the sentence in which the answer can be found. The main idea is that the answer to a given question will probably occur in sentences that contain a rewrite of the original question. For example, given the question *"Who painted the Birth of Venus?"*, a possible rewrite is *painted the Birth of Venus*, which is very likely to appear after the answer *Botticelli*. However, there is also a strong possibility that there are words separating the rewrite and the answer. If we find these sequences, we are able to create patterns that will allow us to find the answers to similar questions. For instance, *ANSWER, who REWRITE* is a pattern that can be extracted from the sentence *Botticelli, who painted the Birth of Venus*.

In the QA track of the TREC-10, the winning system – described in [16] – presents an extensive list of surface patterns and draws the attention of the community to the potential of this technique. Posterior work of [1] details a pattern-learning algorithm, that can be summarized in the following: first, a question (part of it) and its answer are submitted to Altavista; second, the 1000 top documents are downloaded and those containing both the answer and the question are retained; finally, the longest matching substrings are extracted and the question and the answer are replaced by tokens

<QUESTION> and <ANSWER>. Our pattern-learning algorithm is similar to this, although we accept patterns that do not match both the question and the answer. Moreover, our patterns are syntactically-based. Somehow related with the work of [1] is the work of [17] and [18]. The former searches for possible answers in snippets by analysing substrings that have similar contexts of already known answers and uses genetic algorithms in the process; the later bases the performance of the QA system AskMSR on manually created rewrite rules which are likely substrings of declarative answers to questions. The authors also felt the need to produce less precise rewrites, since the correct ones did not match any document. On the Dutch language, [19] explores the question rewriting process for questions that have as answer type *person* and *location*. The authors use syntactic information in the question analysis, but the rewrite rules are hand built, like in [20], which presents an extensive list of regular expressions.

6 Conclusions and Future Work

Here we presented THE-MENTOR, a system that automatically generates multiple-choice test items, composed by a question, a correct answer and a set of distractors. First, it exploits the redundancy of large corpora sources to bootstrap frequent patterns. Each pattern is assumed to bridge the gap between a question and its answer. Afterwards, given a target document, it extracts question/answer pairs from the sentences that match the patterns, as well as the distractors in their surroundings, and builds test items.

By using mainly syntactic information complemented by verb conjugation and WordNet information, the approach we described allowed us to achieve an set of patterns that, after applied to a medium sized target document, could generate a large amount of question/answer pairs, most of which can be used without or after some revision.

As future work, we intend to generate patterns using semantic features, rather than only lexico-syntactic ones. Moreover, we would like to evaluate this approach in texts of different nature. To use dependency grammar is also in our plans in order to allow the system to learn long distance dependencies. When it comes to distractor extraction, we are considering using other sources besides the target document.

Acknowledgments

Ana Cristina Mendes is supported by a PhD fellowship from Fundação para a Ciência e a Tecnologia (SFRH/BD/43487/2008).

References

1. Ravichandran, D., Hovy, E.: Learning surface text patterns for a question answering system. In: ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Morristown, NJ, USA, Association for Computational Linguistics (2002) 41–47
2. Petrov, S., Klein, D.: Improved inference for unlexicalized parsing. In: Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference, Rochester, New York, Association for Computational Linguistics (2007) 404–411

3. Judge, J., Cahill, A., van Genabith, J.: Questionbank: creating a corpus of parse-annotated questions. In: ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, Morristown, NJ, USA, Association for Computational Linguistics (2006) 497–504
4. (omitted for blind review purposes)
5. Li, X., Roth, D.: Learning question classifiers. In: Proceedings of the 19th international conference on Computational linguistics, Morristown, NJ, USA, Association for Computational Linguistics (2002) 1–7
6. Bies, A., Ferguson, M., Katz, K., Macintyre, R., Contributors, M., Tredinnick, V., Kim, G., Marcinkiewicz, M.A., Schasberger, B.: Bracketing guidelines for treebank ii style penn treebank project (1995)
7. Mitkov, R., Ha, L.A.: Computer-aided generation of multiple-choice tests. In: Proceedings of the First Workshop on Building Educational Applications using Natural Language Processing, Edmonton, Canada (2003)
8. Mitkov, R., An Ha, L., Karamanis, N.: A computer-aided environment for generating multiple-choice test items. Nat. Lang. Eng. **12** (2006) 177–194
9. Fellbaum, C., ed.: WordNet: An Electronic Lexical Database. MIT Press (1998)
10. Karamanis, N., Ha, L.A., Mitkov, R.: Generating multiple-choice test items from medical text: a pilot study. In: INLG '06: Proceedings of the Fourth International Natural Language Generation Conference, Morristown, NJ, USA, Association for Computational Linguistics (2006) 111–113
11. Hoshino, A., Nakagawa, H.: Webexperimenter for multiple-choice question generation. In: Proceedings of HLT/EMNLP on Interactive Demonstrations, Morristown, NJ, USA, Association for Computational Linguistics (2005) 18–19
12. Sumita, E., Sugaya, F., Yamamoto, S.: Measuring non-native speakers' proficiency of english by using a test with automatically-generated fill-in-the-blank questions. In: EdAppsNLP 05: Proceedings of the second workshop on Building Educational Applications Using NLP, Morristown, NJ, USA, Association for Computational Linguistics (2005) 61–68
13. Hoshino, A., et al.: A real-time multiple-choice question generation for language testing - a preliminary study. In: Proceedings of the 2nd Workshop on Building Educational Applications Using NLP. (2005) 17–20
14. Pino, J., Heilman, M., Eskenazi, M.: A selection strategy to improve cloze question quality. In: Proceedings of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains. 9th International Conference on Intelligent Tutoring Systems. (2008)
15. Chen, C.Y., Liou, H.C., Chang, J.S.: Fast: an automatic generation system for grammar tests. In: Proceedings of the COLING/ACL on Interactive presentation sessions, Morristown, NJ, USA, Association for Computational Linguistics (2006) 1–4
16. Soubbotin, M.M.: Patterns of potential answer expressions as clues to the right answers. In: TREC. (2001)
17. Figueroa, A.G., Neumann, G.: Genetic algorithms for data-driven web question answering. Evol. Comput. **16** (2008) 89–125
18. Brill, E., Dumais, S., Banko, M.: An analysis of the askmsr question-answering system. In: EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing, Morristown, NJ, USA, Association for Computational Linguistics (2002) 257–264
19. Hoekstra, A., Hiemstra, D., van der Vet, P., Huibers, T.: Question answering for dutch: Simple does it. In: Proceedings of the 18th BeNeLux Conference on Artificial Intelligence (BNAIC), Maastricht, BNVKI (2006)
20. Keselj, V., Cox, A.: Daltrec 2004: Question answering using regular expression rewriting. In: TREC. (2004)