Bootstrapping Multiple-Choice Tests

Sérgio Curto
L²F/INESC-ID
IST, Technical Univ. Lisbon
sslc@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 our approach to automatically generate multiple-choice tests. In a first offline step, we bootstrap a set of lexico-syntactic patterns by taking a set of 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 possibly 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).

Example: Multiple-choice test item

Q. “What is the largest ocean?”

1. Atlantic (distractor)
2. Pacific (correct answer)
3. Indian (distractor)
4. Arctic (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 present our approach to automatically generate multiple-choice tests. We hypothesize that the process of generating multiple-choice tests can rely on the bootstrap of a set of question/answer seed pairs. Here, we present a system that generates multi-choice tests about a free text document, composed by two main components that perform the following tasks:

Bootstrapping lexico-syntactic patterns: this step is performed off-line. A set of question/answering 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 following problem: if patterns are too specific (strong...
patterns), they will not match much and not many tests will be generated; if patterns are too generic (weak patterns), the quality of the tests decreases.

**Generation of test items:** this step runs on-line. 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 – named entities that comply with the question category – are extracted from the same text.

**Evaluation:** the quality of the generated test items is finally evaluated by the user, through a web interface.

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

2 Background

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 (Mitkov et al., 2006). Authors present a system for generating multiple-choice tests that heavily relies on natural language processing techniques and resources, and that is 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 (Fellbaum, 1998). 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 (Karamanis et al., 2006).

Authors (Hoshino and Nakagawa, 2005) and (Sumita et al., 2005) 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 (Hoshino and et al., 2005). 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 learning line, several systems, like REAP (Pino et al., 2008) and FAST (Chen et al., 2006), 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 al-
low 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 (Soubbotin, 2001) – presented an extensive list of surface patterns and draw the attention of the QA community to the potential of this technique. Posterior work of (Ravichandran and Hovy, 2002) details a pattern-learning algorithm, that can be summarized in the following: first, a question (part of it) and its answer is submitted to Altavista; second, the 1000 top documents are downloaded and only the ones 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>. As we will see, our pattern-learning algorithm is similar with this one, 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 (Ravichandran and Hovy, 2002) is the work of (Figueroa and Neumann, 2008) and (Brill et al., 2002). The former searches for possible answers in the 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 a question answering system – AskMSR – on manually created rewrite rules which are likely substrings of declarative answers to questions. These authors also felt the need to produce less precise rewrites, since the correct ones did not match any document. On the Dutch language, (Hoekstra et al., 2006) explores the question rewriting process for questions that have as answer type person and location. These authors use syntactic information in the question analysis, but the rewrite rules are manually built, like in (Keselj and Cox, 2004), which presents an extensive list of regular expressions.

3 Pattern Learning

In this section we detail the pattern learning process.

3.1 General Algorithm

The developed algorithm for pattern retrieval is based on the bootstrapping technique presented in (Ravichandran and Hovy, 2002), and involves the following two stages. First, we use a seed pair – composed by a natural language question, and its respective correct answer – to bootstrap patterns that relate questions and answers. We call B-PATTERNS to the patterns extracted from the bootstrap process. Second, B-PATTERNS are validated using a validation pair – also a natural language question, and its respective correct answer – as input. Therefore, each seed has a validation pair associated. These pairs are automatically obtained, having in mind that although their constituents are lexically distinct, they must share the same syntactic structure as well as the same category. For instance, the seed “Who is the President of France?” cannot be validated by the question “What is the capital of France?”, since both share the same syntactic structure (WHNP VBZ NP), but 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).

In what regards the syntactic analysis of questions, we use the Berkeley Parser (Petrov and Klein, 2007) trained on the QuestionBank (Judge et al., 2006), a treebank of 4,000 parse-annotated questions. In what concerns the question category, we use Li and Roth’s two-layer taxonomy (Li and Roth, 2002), which is one of the most widely known taxonomies for question classification, consisting of a set of six coarse-grained categories and fifty fine-grained ones.

3.1.1 Finding patterns

Algorithm 1 details the first stage in finding B-PATTERNS. It starts by generating permutations of the 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. The wildcard 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
Algorithm 1 Question rewrite patterns bootstrapping algorithm

```
procedure PATTERN-BOOTSTRAP(seed : question-answer pair)
    b-patterns ← []
    phrasal-nodes ← GET-PHRASAL-NODES(seed.question.parse-tree)
    for each permutation in PERMUTE({phrasal-nodes, *, pair.answer}) do
        query ← ENCLOSE-Doubles-QUOTES(permutation)
        results ← SEARCH(query)
        for each sentence in results.sentences do
            if MATCHES(sentence, permutation) then
                B-pattern ← REWRITE-AS-PATTERN(permutation, phrasal-nodes)
                B-patterns ← ADD(B-patterns, B-pattern)
            end if
        end for
    end for
    return B-patterns
end procedure
```

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 (Ravichandran and Hovy, 2002), 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?”, for which the parse trees and phrasal nodes are depicted in Figure 1. It does not make sense to divide the noun-phrase the Birth of Venus, since that 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 results 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 “[WHNP Who] [VBD painted] [NP the Birth of Venus]”, and suppose the sentence Botticelli has painted the Birth of Venus matches the permutation Botticelli * painted the Birth of Venus. The resulting pattern would then be “\{ANSWER\} has VBD NP”, which is created by replacing each phrasal node with the respective tag, and the seed answer with the tag “\{ANSWER\}”.

3.1.2 Pattern Validation

While many of the patterns generated by Algorithm 1 are generic enough to be applied to other questions, there are others specific to the seed question-answer pair. For instance, consider 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 other 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 question-answer validation pair.

The validation algorithm works by testing each generated pattern against the validation pair, and calculating its precision, which is the ratio between the number of times the pattern correctly matched and the number of times the pattern was expected to match. We measure this as the maximum number of results retrieved by the search engine. For example, using the previously mentioned 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, which would cause the pattern to

\(^1\)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.
be ruled-out, as the algorithm dictates that each pattern must have precision larger than some threshold in order to be retained.

3.2 Handling Verbs

The aforementioned method 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 behavior for Noun- and Prepositional-phrases that are needed to 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 oughts be flexible enough to capture a pattern in the sentence Boticelli 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).

We call inflected to the B-PATTERNS generated by verb inflection.

3.3 Allowing Weak Patterns

There are, however, some patterns that should not be disregarded, although they do not contain all question phrases and cannot be handled by allowing verbs’ multiple inflexion. These patterns arise from the passages 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 Boticelli, even if it does not include the verb (in this case, painted). These patterns need to be differentiated from strong and inflected patterns, not only because of the way they were created, but also because they will trigger distinct strategies in the test item generation. Therefore, we will call weak patterns to the B-PATTERNS that do not contain all phrases of the seed questions and are not generated by verb inflection. To generate weak patterns, the procedure is similar as referred, just we do not allow Verb Phrases in the question to be present on the permutations.

Table 1 shows a set of patterns2 generated from, and validated with, the passages retrieved by the search engine, for questions with flatten syntactic structure WHNP VBD NP and category HUMAN:INDIVIDUAL.

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

<table>
<thead>
<tr>
<th>Question</th>
<th>HUMAN:INDIVIDUAL-WSHP VBD NP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>0.625</td>
<td>{ANSWER}'s NP</td>
</tr>
<tr>
<td>0.25</td>
<td>{ANSWER} began VBG NP</td>
</tr>
<tr>
<td>0.625</td>
<td>NP VBD by {ANSWER}</td>
</tr>
</tbody>
</table>

2Here, as well as throughout the entire paper, the Penn Treebank II Tags (Bies et al., 1995) are used.
4 Building Multiple-Choice Tests

The next stage in the generation of multiple-choice test items is done online. The user specifies the target documents and the system parses the texts and applies the previously learned patterns in order to obtain question/answer pairs (as well as distractors). This method also involves several strategies for filtering the obtained question/answer pairs in order to discard low quality pairs.

4.1 Extracting Question/Answer Pairs

Our method relies on matching lexico-syntactic information from the B-PATTERNS against the parsed sentences of the target document. Each match is done at different 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 to find all the occurrences of a given pattern on the syntactic tree of the parsed sentences. This algorithm is, however, out of the scope of this paper.

4.2 Filtering

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. In this section we report some of the filters in use.

4.2.1 Forcing Question/Answer Category Matching

Since the question and its B-PATTERNS were grouped according to the question’s category, that represents the semantic class of the expected answer, extracted fragments in which the answer does not comply with the expected category can be discarded. Consider, for example, the question and answer pair: “Who was François Rabelais?”-An important 16th century writer. Clearly, the answer is in accordance to the semantic class expected by the question (HUMAN:INDIVIDUAL), which is indicated by the word writer.

Being so, 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. In order to do so, we have 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.

This strategy has, however, the problem of not considering the answer as a whole, rather the words it is composed of. For instance, in the next sentence Eiffel, the french engineer and after whom the tower is named, validates the answer: “Who is an icon of both Paris and France?”- The Eiffel Tower

4.2.2 Discarding Anaphoric References

We rely on a a group of simple regular expressions to invalidate questions that contain anaphoric references and others, which we empirically know that will not result in quality multiple-choice test items. Therefore, questions such as “What is it?”, “Where is there?” or “What is one?” are discarded.

4.3 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. For all the components that do not
appear in the fragment, the components in the question is used.

In what concerns distractors’ generation, we search in the text for phrases that have the same category of the question. Then we chose the ones that are near the sentence that originated the question/answer pair, but that are not in that same sentence. As an example, if we consider the sentence *This resource briefly explores the telegraph invented by Samuel Morse*, that originates the question/answer pair “Who invented the telegraph”-Samuel Morse, as its category is HUMAN:INDIVIDUAL, we will search for other HUMAN:INDIVIDUALS in the nearer sentences.

Figure 2 illustrates the whole process of multiple-choice test items generation.

5 Experiments
This section describes the experiments we conducted to evaluate our system.

5.1 Bootstrapped Patterns
The presented approach takes as input natural language questions and their correct answers. In our experiments, we used 206 natural language question/answer pairs, some taken from an on-line trivia, others manually created.

All questions are factoids and pertaining to the following 10 different categories – ENTITY:CURRENCY (#Q/A pairs: 10), ENTITY:SPORT (2), ENTITY:LANGUAGE (8), HUMAN:INDIVIDUAL (34), LOCATION:.City (22), LOCATION:COUNTRY (33), LOCATION:STATE (13) and NUMERIC:DATE (40).

These pairs were then automatically combined and grouped (according with their category and syntactic structure), in order to create the seed/validation pairs.

This preprocessing step resulted in a set of 668 seed/validation pairs, which, along with the different syntactic structures and categories, 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.

Table 2: Example of seed-validation pairs.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td>LOCATION:COUNTRY-WHPP VBD NP VBN</td>
<td>S “In which country was Bjorn Borg born?”-Sweden</td>
<td>V “In which country was the match invented?”-France</td>
</tr>
</tbody>
</table>

The 16 top ranked snippets retrieved by the web search engine Google were used to learn the B-PATTERNS, according with the process described in Section 3. This resulted in 1348 patterns, from which 1126 were unique.

The distribution of patterns according to the category is presented in Figure 3.
As it can be seen, the more seed/validation pairs, the more B-PATTERNS were bootstrapped by category. However, an exception to this was the category ENTITY:LANGUAGE, for which a small number of pairs (8) 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, although this was not 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%.

5.2 Metrics
To evaluate the generated test items we used a similar model of test item review to that of (Mitkov and Ha, 2003). If the items make no sense they are discarded; they are marked as worthy otherwise. For instance, the item “Who left alone much?” - the new British rulers is discarded. Then each worthy item is evaluated by the degree of review needed, and classified as:

- Minor, if the test item requires minimal corrections (punctuation, article introduction or spelling corrections), for example: “Who was François Rabelais?” - An important 16th century writer
- Moderate, if the test item requires the removal/insertion or reordering of words, or if a set of distractors are not applicable to the question item, for example, in: “Who was Eugène Viollet-le-Duc?” - the associated architect the answer should have the words “the associated” removed;
- Major, if the test item 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?”.

5.3 Evaluation
To evaluate the generation of test items we used the Wikipedia article about the “History of Portugal” as target document. Results for each type of pattern – strong, inflected and weak – are shown in Table 3.

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Degree of Review</th>
<th>Discarded</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min.</td>
<td>Mod.</td>
<td>Maj.</td>
</tr>
<tr>
<td>S</td>
<td>14</td>
<td>26</td>
<td>11</td>
</tr>
<tr>
<td>I</td>
<td>2</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>W</td>
<td>0</td>
<td>2</td>
<td>55</td>
</tr>
<tr>
<td>Total</td>
<td>16</td>
<td>35</td>
<td>74</td>
</tr>
</tbody>
</table>

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 question/answer pairs. Although more patterns were activated, they did not succeed to pass the filtering phase of the item generation process.

As expected, most of the question/answer pairs extracted from the strong patterns generated items considered as worthy. These 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), mainly using strong pattern matches. 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;
- **Location:Other**: generated 18 pairs. Since this category is more generic that the previous...
two Locations, it could generate better question items than the others (for instance, a question/answer 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 were matched for the other four categories, 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 other (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, they were mostly appropriate to the generated question 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.

### 5.4 Discussion

Our approach receives as input a set of natural language question/answer pairs and generates question items in order to create multiple-choice tests. With our this approach, several question items were generated automatically that can help the creation of multiple-choice tests, originated from a small set of question/answer 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 question items generated by them: STRONG patterns generated better question/answer pairs, however in a lower number, and WEAK patterns generated lower quality question/answer pairs, 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 question 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 question/answer pairs and the automatic filtering phase should be improved.

Also, it mostly 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 question items, most of which can be used.

### 6 Conclusions

Here we presented an approach to automatically generate multiple-choice test items, composed by a question, an answer and a set od distractors. First, it exploits the redundancy of large corpora sources in order 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 and answer pairs from the sentences that match the patterns, as well as the distractors in their surrounding, 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.
References


Martin M. Soubbotin. 2001. Patterns of potential answer expressions as clues to the right answers. In TREC.