INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Automatic generation of multiple-choice tests
Geração automática de testes de escolha múltipla

Sérgio dos Santos Lopes Curto

Dissertation for obtaining the Master's Degree in
Information Systems and Computer Engineering

Jury

President:	Doutora Maria dos Remédios Vaz Pereira Lopes Cravo
Advisor:	Doutora Maria Luísa Torres Ribeiro Marques da Silva Coheur
Co-advisor:	Doutor Nuno João Neves Mamede
Evaluation Jury:	Doutor Jorge Manuel Evangelista Baptista

September 2010

Acknowledgements

I would like to thank my advisors, Professor Luisa Coheur and Professor Nuno Mamede, for their guidance, discussion and motivation.

Thanks also Daniel Santos for is availability on the resolution of problems with integration of his work with mine and special thanks to Ana Mendes for all the assistance provided.

I also want to thank all the members of L2F, that provided me with essential tools and discussion of solutions.

Thanks to my family for believing in me and providing all the support each day.

Finally, I would like to thank all my colleagues that worked with me on Instituto Superior Técnico, with special attention to Rita Santos, Henrique Rodrigues, André Costa and Carlos Pona.

Lisbon, October 25, 2010

Sérgio dos Santos Lopes Curto

To my parents, Francisco and Helena,

To my brother, Pedro,

Resumo

Testes de escolha múltipla são uma forma rápida de avaliar conhecimento. No entanto, a criação manual de testes de escolha múltipla é um processo difícil e demorado. Nesta tese propomos a criação automática destes testes.

Geradores de testes de escolha múltipla têm dois componentes principais. O primeiro é responsável pela geração dos pares pergunta/resposta, o segundo pela geração de distractores. Neste trabalho, são seguidas duas abordagens:

1. A primeira é baseada em regras que utilizam as dependências e os traços das palavras, identificados por uma cadeia de processamento de língua natural, a fim de criar pares pergunta/resposta e distractores, respectivamente. Estas dependências e regras são criadas manualmente;
2. A segunda usa a web para gerar automaticamente padrões dado um conjunto de sementes (pares pergunta/resposta). Neste processo, são gerados um conjunto de padrões sintácticos e utilizados para criar testes de escolha múltipla. Vários filtros são aplicados.

Adicionalmente, foi desenvolvida uma interface para o sistema automatizado e ambas as abordagens foram testadas com corpus de jornais e/ou textos da Wikipédia. Os resultados mostram que a maioria dos testes de escolha múltipla gerados por ambas as abordagens podem ser usados com pequenas alterações.

Abstract

Multiple-choice tests are a fast way to evaluate knowledge. However, manually building quality multiple-choice tests is a difficult and time consuming process. In this thesis we propose to automatically build such tests.

Multiple-choice tests generators have two main components. The first is responsible for the generation of question/answer pairs; the second for the generation of distractors. In this work, two approaches are followed:

1. the first approach is based on rules that use dependencies and word features, identified by a natural language processing chain, in order to return question/answer pairs and distractors, respectively. These dependencies and rules are manually created;
2. the second approach uses the web to automatically generate patterns given a set of seeds (question/answering pairs). In this process, a set of syntactic patterns are generated and used to create multiple-choice tests. Several filters need to be applied.

Additionally, an interface for the automated system was developed and both approaches were tested in a newspapers corpus and/or in wikipedia texts. Results show that the majority of the multiple-choice tests returned by both approaches can be used with minor changes.

Palavras Chave
Keywords

Palavras Chave

Geração de exames de escolha-múltipla assistida por computador

Multilíngue

Descoberta de padrões

Geração Automática de Questões

Geração Automática de Distractores

Keywords

Computer Aided Multiple-Choice Test Items Development

Multilingual

Pattern Discovery

Automatic Questions Generation

Automatic Distractors Generation

Contents

List of Figures

List of Tables

Acronyms

BNF Backus-Naur Form

CAID Computer-aided multiple-choice test items development

FAST Free Assessment of Structural Tests

NER Name Entity Recognition

NLP Natural Language Processing

POS Part of Speech

QA Question-Answering

QG Question Generation

REAP Reader-Specific Lexical Practice for Improved Reading Comprehension

XIP Xerox Incremental Processor

XML Extensible Markup Language

1 Introduction

1.1 Motivation

Multiple choice tests are seen as a good way to evaluate students knowledge, covering several topics in a short amount of time. Each test item is composed by a question and a group of answers, in which only one is correct. Incorrect answers are called distractors.

Example 1: multiple choice test item

What is the name of the grammatical class that denotes an occurrence or situation?

- Name (distractor)
- Verb (correct answer)
- Article (distractor)
- Substantive (distractor)

■

The creation of multiple choice tests can become a time-consuming, trial and error process, until proper questions are created. To this end, computer aided multiple-choice tests generation, that this thesis focus on, can help reducing the time needed to make several different tests. This process can be divided into:

1. Question/Answer generation;
2. Distractor selection.

In order to generate a question/answer pair there has to be a pool of information about a certain topic. This source can be, for instance, a book, a summary or an on-line wiki. After the source selection, sentences are processed in order to generate questions and the respective correct answer. As an example, consider the following sentence:

Example 2: sentence

A dictionary is a compilation of words. ■

This sentence can be converted into the following pair question/correct answer:

Example 3: pair question/correct answer

What is a compilation of words?

- A dictionary (correct answer)

■

The selection of distractors is an important step on multiple choice question generation, since poorly selected distractors can lead to an easy identification of the correct answer, for example:

Example 4: multiple choice test item with bad distractors

What is a compilation of words?

- A car
- A dictionary
- The woods
- The storm

■

In this example we can easily identify that the option "A dictionary" is the only answer that can be selected for a question that asks something about words, so this multiple-choice test item would be very poor to evaluate knowledge.

Another thing that needs to be taken into consideration for the success of distractor selection process is that the selected distractors must be related to the correct answer, not only in subject but also, for instance, in gender.

1.2 Goals

The goal of this thesis is to create a platform to assist on multiple choice test items generation based on input corpus. This corpus can be given by user or selected from the available wikipedia articles.

On the question generation, the platform is pattern based and support the discovery of new patterns. On the distractor generation, it explores the corpus which was used for question generation, in order to retrieve distractor candidates belonging to the same context.

The platform presents an user interface for review and edition of the multiple-choice question items generated by the automated pattern discovery system, seeking to assist usage of the system.

1.3 Document Overview

This document starts by presenting related work on Chapter 2, followed by the presentation of two developed systems one supported by manually created rules on Chapter 3 and another based on bootstrapped rules on Chapter 4. It will end with the conclusions on Chapter 5.

2 State of the art

2.1 Introduction

This state of the art explores the current research on computer multiple-choice test generation and related work. The main focus are the different projects that are under development with the goal of generating multiple-choice question items (Section 2.2). Nevertheless, several frameworks on question generation are described (Section 2.3) and possible distractor sources are analyzed (Section 2.4).

2.2 Systems

The systems presented in this section share some common tasks. As so, in order to improve readability, these systems's description is organized by the following basic tasks:

- Question generation - The question generation task is one of the core tasks of all question item generation systems. In this task, sentences on provided corpus are selected and a question/correct answer pair is generated.
- Distractor generation - During this task, distractors are generated based on different criteria. To this end different methods of distractor selection and/or sources are analyzed.
- Test item review and finalization - Some systems require these steps to accomplish the test item generation process after question and distractors generation. Some systems require manual user intervention to finish the question item generation process while others automatize the process completely.

2.2.1 Computer-aided multiple-choice test items development

Computer-aided multiple-choice test items development[1] (Mitkov et al., 2006) (Mitkov & Ha, 2003) (CAID) is a system for generation of multiple-choice test items based on natural language processing

[1] http://clg.wlv.ac.uk/research/current-projects.php - Visited on December 2009

created by the *Research Group in Computational Linguistics's* of *University of Wolverhampton* (United Kingdom).

The system is focused on the following principles:

- multiple choice test should focus on key concepts in order to evaluate the correct topics;
- distractors should be semantically close to the correct answer to avoid giving clues to discard distractors.

2.2.1.1 Question generation

Terms are extracted from the input text to be used as anchors in question generation and distractor selection. To this end, nouns are identified and the system stores the number of times each was detected. The most frequent ones that agree with a group of regular expressions are considered to be terms.

After terms extraction, sentences that contain domain specific words or with certain structures are selected and a set of question generation rules are applied. For example, after detecting the sentence "A dictionary is a compilation of words with the respective meaning." the system generates the question "What is a compilation of words with the respective meaning?".

The generated questions are then filtered by agreement rules assuring that they are grammatically correct and then gender-specific rules remove sentences that refer examples, tables or previous mentioned entities.

2.2.1.2 Distractor generation

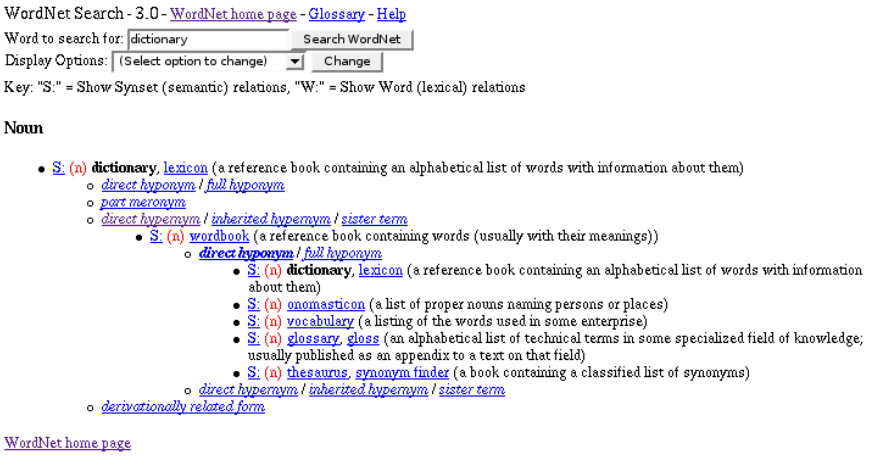
WordNet is searched for semantically alike concepts close to the correct answer, with preference to words that are on the corpus. Search results are marked as possible distractors.

For example, considering the previous sentence "A dictionary is a compilation of words with the respective meaning.", the system would search WordNet for distractors to "dictionary". With the search presented on Figure 2.1, "vocabulary", "glossary" and "thesaurus" would be selected as distractors.

2.2.1.3 Test item review and finalization

In this last step, CAID uses a manual item revision process. Each question is reviewed by the user and the system presents the sentence that originated the question, the generated question, a group of selected distractors and a group of alternative distractors.

Figure 2.1: WordNet search for distractors of "dictionary"



The user starts by accepting or rejecting the generated test item depending on the degree of review needed. When a question is accepted, the user manually corrects the test item and marks the question with the following classification, depending on how much work is needed to correct it:

- Minor - test item only required minimal corrections on the structure of the generated question (punctuation, article introduction or spelling corrections);
- Moderate - test item required removal or reordering of words and/or replacement of up to one distractor;
- Major - test item required a deep grammatical correction and/or replaced two or more distractors.

From a total of 575 test items generated, 57% were considered for further use and from this group 6% were used without any modification and the remaining ones subject to post-editing. Considering the test items that required post editing 17% required minor revision, 36% moderate revision and 47% required major revision.

Two graduate students in linguistics used both CAID and manual generation of test items. With CAID, 300 test items were ready to be used requiring an average of 1 minute and 48 seconds to generate each. Without CAID 65 test items were generated manually with an average of 6 minutes and 55 seconds to generate each test item.

2.2.2 WebExperimenter

WebExperimenter (Hoshino & Nakagawa, 2005) is another system for multiple-choice test items generation, under development by the *Interfaculty Initiative in Information Studies* of *University of Tokyo*.

2.2.2.1 Question generation

This system generates test items with blank spaces, which are to be filled in by the selected option[2].

Example 5: cloze question

A [ ] is a heavy vehicle that can be used to transport goods by land.

1. Truck
2. Automobile
3. Boat
4. Airplane

■

The system receives an input text that is processed by an instance converter that generates possible questions with blank space.

Example 6: instance generation

With the following input text "A truck is a heavy vehicle that can be used to transport goods by land." the following instances are generated:

- [ ] truck is a heavy vehicle that can be used to transport goods by land.
- A [ ] is a heavy vehicle that can be used to transport goods by land.
- A truck [ ] a heavy vehicle that can be used to transport goods by land.
- ...
- A truck is a heavy vehicle that can be used to transport goods by [ ].

■

Afterward the system processes a set of pre-built questions, using semi-supervised learning[3] in order to detect instances that are not valid for question generation. Instances marked as invalid are discarded and valid ones are selected.

2.2.2.2 Distractor generation

This system suggests that the user can choose the distractor's source between WordNet, Edit Distance, Mutual Information or randomly chosen source, yet it does not specify how those can be used in order to produce the distractors.

[2] Also known as cloze questions.

[3] Class of machine learning techniques that make use of both labeled and unlabeled data for training - typically a small amount of labeled data with a large amount of unlabeled data. Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). (from en.wikipedia.org)

2.2.2.3 Test item review and finalization

WebExperimenter does not present any form of post-generation test item review as all the test items are generated automatically by the system. Also, unlike CAID, no evaluation was presented on the articles found.

2.2.3 Web application for question generation assistance

Web application for question generation assistance (Hoshino & Nakagawa, 2007) is another system for multiple-choice test items[4]. Sharing the same authors with the WebExperimenter, this system approaches the generation of questions by a different perspective. The system is developed in order to assist the learning of English as second language and uses on-line news articles as source text, in order to retrieve a well written and up-to-date corpus. The system is composed by a pre-processor program and a user interface.

During the pre-processing phase, the system updates the news articles available downloading them directly from the targeted web-sites and performs all the automated tasks (text parsing, question generation and distractor selection).

The user interface component allows the generation of the question items using all preprocessed corpus.

2.2.3.1 Question generation

Unlike WebExperimenter the generation of questions is semi-automated, in order to improve the quality of the questions generated, since learning a new language with the help of multiple-choice questions requires high quality question items.

The question generation is present on two phases, during the pre-processing of the downloaded text and later during the user interaction.

As so, in the pre-processing phase, the system starts by parsing the text, followed by the selection of question targets. The system generates blank space questions just like WebExperimenter, using semi-supervised learning.

Later during the user interaction phase, the user is presented with the generated questions for the selected news articles and manually selects the desired questions.

[4] under development by the *Interfaculty Initiative in Information Studies* of *University of Tokyo*

Table 2.1: Example of transformation rule to create grammar distractors

Grammar target	Matching Rule	Generation rules for wrong alternatives
to + infinitive	(S (VP (TO to) (VP (VV) ...)))	to do -> do; to do -> doing; to do -> done
Legend for matching rule: A pair of brackets indicates a phrase. The sign on the left side of the bracket indicates the part of speech assigned to the phrase. S: Sentence, VP: Verb Phrase, TO: preposition "to", VV: verb (infinitive)		

2.2.3.2 Distractor generation

In what concerns the distractor generation, in the pre-processing phase the system generates vocabulary distractors and grammar distractors.

Vocabulary distractors are related words with a frequency of occurrence close to that of the right answer or to synonyms of it. In order to improve the quality of the distractor generation, the system selects words with the same part of speech and inflection as the right answer.

Grammar distractors are words generated by the use of ten grammar targets. For each target the system applies a pattern that matches a phrase with a specific structure and follows three generation rules which transform the matched phrase into possibly grammatically incorrect rules (see table 2.1).

For example, when the first pattern matches "to go to the theater", in the sentence "We are planning to go to the theater", the system would generate the following grammar distractors suggestions:

- go to the theater
- going to the theater
- went to the theater

On the second phase, during user interaction, after the user selected one of the highlighted parts of the text in order to generate a question, the system presents suggestions of vocabulary and grammar distractors and the user selects one of each type.

At the end of the distractor generation process, the system generates to each question a grammar and vocabulary distractor. This last distractor is determined by a combination of the selected suggestion for both vocabulary and grammar distractors.

For example, given the sentence "We are planning to go to the theater", if the user creating the question selects the grammar distractor "going to the theater" and the vocabulary distractor "arrive" to replace "go", the system would generate the following multiple choice question item:

Example 7: Web application multiple choice question item

We are planning ...

- to go to the theater (correct answer)
- going to the theater (grammar distractor)
- to arrive to the theater (vocabulary distractor)
- arriving to the theater (grammar and vocabulary distractor)

■

2.2.3.3 Test item review and finalization

The review and finalization component was described on the question and distractor generation sections of the system, as it follows a semi-automated procedure to generate the test items.

Once all test items are generated, the user prints the test item list and a complete multiple-choice exam developed with the support of the system is presented.

2.2.4 REAP

REAP (Brown et al., 2005) is also a system to assist the learning of a language. The system presents the user with texts according to his reading levels. To this end the system constructs an accurate model of vocabulary knowledge using it to find documents that have some words known by the student and some to be learned.

After reading the new text, the student understanding of the new words is evaluated and the model of vocabulary knowledge is updated, depending on the answers given by the student. This allows the system to retrieve new texts with new words and the process can be repeated.

There is also a version of this project applied to Portuguese (Marujo et al., 2009), with a automated question generation system (Correia, 2010).

Questions can have two different forms. In a word-bank form the testee sees a list of answer choices followed by a set of questions with blank spaces to fill with the words from the word-bank.

Example 8: REAP word-bank question

Word-bank: verbose, infallible, obdurate, opaque

Choose the word from the word-bank that best completes each phrase bellow:

1. ___ windows of the jail
2. the Catholic Church considers the Pope ___

3. ___ and ineffective instructional methods
4. the child's misery would move even the most ___ heart

■

Another other question form consists in cloze questions, where the testee sees a phrase with a blank space that has to be filled with one of the given words. Only one word is correct and all the other incorrect options are distractors.

Example 9: REAP cloze question

The child's misery would move even the most ___ heart.

1. torpid
2. invidious
3. solid
4. obdurate

■

For this view of the state of the art, we will focus on the automatic question generation components developed by (Correia, 2010). The main lexical resources, focused on word relations, used by this system are:

- Porto Editora's Associated Words - Linguateca (PAPEL[5]);
- MultiwordNet of Portuguese (MWN.PT[6]);
- TemaNet[7].

The system focus on generating test items for a list of words that the student should learn. The Portuguese version used is called Portuguese Academic Word List (Baptista et al., 2010) (P-AWL).

2.2.4.1 Question generation

The automated question generation component creates the following types of questions:

- definitions - requires a definition of a word. Based on the definitions present on TemaNet and Porto Editora's online dictionary;

[5]http://www.linguateca.pt/PAPEL (last visited on October 2011
[6]http://mwnpt.di.fc.ul.pt/features.html#main (last visited on October 2011
[7]http://www.instituto-camoes.pt/temanet/inicio.html (last visited on October 2011)

- synonym, hyperonym and hyponym - relations between words with same meaning, class of instances to instance members and instance members to class of instances, respectively. Based on the relations between words on PAPEL, MWN.PT and TemaNet;

Cloze questions are also used on this system but they are selected from a set developed at *Universidade of Algarve*. These questions require the use of a word to complete a sentence or a phrase.

On the definition questions, since words usually have more then one possible sense and since no disambiguation mechanism is present, the first sense found in the dictionary is used which is usually the most common sense. If the dictionary does not have a specific word present, the system attempts to retrieve the definition from TemaNet.

An example of a definition question presented by (Correia, 2010) is:

Example 10: REAP.PT - definition question/answer pair
Qual das seguintes é uma definição da palavra *criar*?

- conceber algo original

■

The synonym, hyperonym and hyponym questions are based on analyzing the relations available for the target word on different lexical resources. Relations present in two or more resources are preferred over the ones present in only one resource.

An example of a synonym question presented by (Correia, 2010) is:

Example 11: REAP.PT - synonym question/answer pair
Precisamente é um sinónimo de ___

- exactamente

■

2.2.4.2 Distractor generation

On definition questions, distractors are generated by randomly using definitions of the remaining P-AWL words. Completing the example presented before, the system generates the following question item:

Example 12: REAP.PT - definition test item
Qual das seguintes e uma definição da palavra criar?

- limpar com substâncias que evitem a propagação de agentes infecciosos

- enunciar, uma a uma, todas as partes de um todo
- conceber algo original
- apresentar ampla e detalhadamente

■

On synonym, hyperonym and hyponym questions the system selects words with the same POS of the *focus word* and with same classification level of the answer. Again, completing the example presented the system generates the following question item:

Example 13: REAP.PT - synonym test item
Precisamente é um sinónimo de ___

- arbitrariamente
- fisicamente
- ilegalmente
- exactamente

■

For cloze questions, automated distractor generation creates graphemic and phonetic distractors. On the graphemic distractors, the P-AWL words with same POS and with less then 5 edit distance using Levenshtein Distance (Levenshtein, 1966)[8] are selected, giving preference to the ones with less edit distance.

The phonetic distractors are created by exploring most common spelling errors for Portuguese language (for example "ss" is frequently confused with "ç").

Additionally, on this question type, distractors candidates generated by the previous methods are filtered, using PAPEL and MWN.PT, removing candidates that would otherwise be correct answers although being different from the target word.

2.2.4.3 Test item review and finalization

The REAP system is designed based on continued adaptation between the testee skill level and the presented questions. In order to achieve this target all the questions are generated in order to cover a set of words that the student should learn.

[8] "The Levenshtein distance between two strings is defined as the minimum number of edits needed to transform one string into the other, with the allowable edit operations being insertion, deletion, or substitution of a single character." - from en.wikipedia.org (October 2010)

These words are distributed by learning levels, guiding the learning process of the student. The three lexical resources presented currently cover 81.82% of the P-AWL words and are a core component of the creation and filtering of the test items.

2.2.5 Free Assessment of Structural Tests

Free Assessment of Structural Tests (Chen et al., 2006), FAST, also aims to assist the teaching and learning of English as foreign language. To this end, the system develops two types of test items:

- Cloze multiple-choice test - composed by sentence with blank space and then a list with correct answer and distractors to fill that blank space;
- Error detection test - where a sentence is presented with several words underlined where one of the underlined words is the error to be detected.

An example of an error detection test, from the cited article is:

Example 14: error detection test

Although maple trees are among the most colorful varieties in the fall, they lose its leaves sooner than oak trees. ■

In this example, the word "its" is the error that the testee is expected to detect and the other underlined phrase chunks are distractors.

2.2.5.1 Question generation

In order create tests, FAST requires a manually developed set of patterns and a URL for web-pages that are used as source. It then produces a set of grammar test items for the user.

To create these patterns, sentences with similar structural features are reviewed. Using the example on the article referred "My friends enjoy traveling by plane." and "I enjoy surfing on the Internet." share identical syntactic structure. In this situation the pattern "*enjoy X/Gerund*" would be made. This pattern requires a sentence with the verb "enjoy" needing a gerund as the complement.

The verb "enjoy" can be easily replaced by other verbs such as "admit" or "finish", so we can just write a more generic construct pattern "*VB VBG*", were those two tags are the ones produced by the POS tagger when it analyzes the presented sentences.

2.2.5.2 Distractor generation

Distractor generation patterns are then made for each of the previous construct patterns. Distractors are usually composed by a word matched on the previous pattern with some modifications. For example, in "I just finished writing on my blog." distractors generated to replaced "writing" could be "write", "written" and "wrote", which the POS tagger identifies as VB, VBN and VBD respectively.

For the distractor generation patterns, the system uses $0 to refer to the first word matched by the construct pattern, in the previous example the word "writing", $1 to the second word and so on. So for the previous pattern, found during the question generation phase *VB VBG*, we could create the following distractor patterns *$0 VB*, *$0 VBN* and *$0 VBD* for the example presented.

2.2.5.3 Test item review and finalization

Next, a web crawler extracts texts from several web-sites containing texts of learned genres (textbook, encyclopedia, ...), removing HTML tags and tagging each word with a Part of Speech tag. Then test items are generated:

- Cloze multiple-choice tests - the word selected by the distractors pattern is removed leaving a blank space, then distractors are generated applying the patterns;
- Error detection tests - the target word is selected and replaced by a wrong word distractor and three phrase chunks are chosen as distractors. The testee has to select the chunk that has the replaced word.

To test the system 69 test patterns covering nine grammatical categories were developed and several articles were extracted from Wikipedia and VOA (Voice of American). In order to improve readability and self-contain characteristic of grammar questions, the first sentence of each article and selected sentences based on readability were removed. Then the system matched the patterns with the articles transforming 3 872 sentences into 25 906 traditional multiple-choice questions and 2 780 sentences into 23 221 error detection questions.

From this a large amount of verb-related grammar questions were blindly evaluated by seven professor/students and from a total of 1359 multiple-choice questions 77% were regarded as "worthy" (can be directly used or only require minimal revision), while 80% of the 1908 error detection tasks were deemed to be "worthy".

2.2.6 Lexical information for composing multiple-choice cloze items

In order to support the learning of English vocabulary (Liu et al., 2005), the *Department of Computer Sciences of National Chengchi University* (Taiwan), and the *Department of Foreign Languages and Literatures of National Taiwan University* (Taiwan), developed a system for composing multiple-choice cloze items.

The system presents two core components, the sentence retriever and the distractor generator, which are described on the next sections.

2.2.6.1 Question generation

The question generation is the task of the sentence retriever component. This component starts by extracting sentences from a part of speech tagged corpus that contain certain key part of speech tags, creating target sentences. When such sentences are found, the sense of the key is also checked for requirements.

In order to choose the sense of a key, that word is searched on WordNet for meanings. Given the example presented on the cited article, to find the sense of "spend" in the candidate target sentence "They say film makers don't spend enough time developing a good story." WordNet is searched. With this search, WordNet returns two possible meanings of "spend":

1. spend, pass - pass (time) in a specific way; "How are you spending your summer vacation?"
2. spend, expend, drop - pay out; "I spend all my money in two days."

The first sense has one head word "pass", that summarize the meaning, and a sample sentence for sense. The second sense has two head words "expend" and "drop" and a sample sentence.

In order to find the meaning of "spend", the key on the target sentence is replaced with each head words and the ones that fit better on that sentence determine the sense of "spend".

Once a sentence meaning is found the key is removed and the question generation process ends.

2.2.6.2 Distractor generation

Based on the key extracted from the target sentence selected during question generation, distractors are selected using word frequency in order to avoid uncommon words that may be detected by the testee.

Distractors with the same part of speech are selected, with special attention to Taiwan cultural background. To do so, when ever possible words with similar Chinese translations with the key are assigned to become distractors, since Taiwan students usually find difficult to differentiate English words that have very similar Chinese translations.

2.2.6.3 Test item review and finalization

The system does not present any form of post-generation test item review as all the test items are generated automatically by the system.

Sentences containing verbs, nouns, adjectives and adverbs are chosen, and the word sense of each of those target words are selected on WordNet. The selection of a word sense is dependent on a threshold, and the higher it is, the higher is the accuracy of the chosen sense at the cost of rejecting more sentences. Also, senses for the target word that do not contain any sample sentence are rejected, which leads to the discrimination of these senses.

From 160 sample sentences, the system created an average of 65.5% correct target sentences with the following key item types:

- Verb key - 77 target sentences;
- Noun key - 62 target sentences;
- Adjective key - 35 target sentences;
- Adverb key - 26 target sentences.

Then the test item generator creates 200 items for evaluation using the target sentences mentioned. In the end, an average of one in each two test items presents correct target sentence and good distractors.

2.2.7 System comparison

The presented systems can be compared by the way they approach each of the basic components of question generation. The following table presents the how each system approaches question generation:

Table 2.2: System comparison table - Question Generation

Topic	**CAID**	**Web Experimenter**	**Web Application**	**REAP.PT**	**FAST**	**Lexical Information**
Question/answer multiple-choice questions	X	-	-	X	-	-
Cloze questions	-	X	X	X	X	X
Error detection questions	-	-	-	-	X	-

Then a summary of the technics each system applies on both question and distractor generation:

Table 2.3: System comparison table - Technics

Topic	**CAID**	**Web Experimenter**	**Web Application**	**REAP.PT**	**FAST**	**Lexical Information**
Pattern	QG / -	- / -	- / -	- / -	QG / D	QG / D
Word relation resources	- / D	- / D	- / -	QG / D	- / -	QG / -
Word frequency	- / -	- / -	- / -	- / -	- / -	- / D
Semi-supervised learning	- / -	QG / -	QG / -	- / -	- / -	- / -

QG stands for question generation; D stands for distractor generation;

By last, an analysis of how each system approaches item revision:

Table 2.4: System comparison table - Item revision

Topic	**CAID**	**Web Experimenter**	**Web Application**	**REAP.PT**	**FAST**	**Lexical Information**
Manual	X	-	X	-	X	-
Automatic	-	-	-	X	-	-
None	-	X	-	-	-	X

From the system comparison tables 2.2, 2.3 and 2.4 we can see that each of the systems presented solves question generation in a different way. The purpose of each system is also different, some chose to seek to aid generation of multiple-choice questions (CAID, Web application and FAST systems), while others seek fully automatic question generation (WebExperimenterand, REAP.PT and Lexical information systems).

The sources of distractors usually rely on some kind word similarity (on the corpus or on a pre-compiled source), the use of relations present on lexical resources (English WordNet, PAPEL, MWN.PT and TemaNet), word frequency or edit distance. Some of the systems presented seek to aid the learning of a language and some use user interaction in order to achieve high quality standards.

2.3 Question Generation

Over the years, several workshops and conferences focused on the question generation topic. Among these, we highlight the Workshop on the Question Generation Shared Task and Evaluation Challenge[9] (September 25-26, 2008, National Science Foundation, Arlington, United States of America) which is aimed at establishing a community consensus with respect to a shared task evaluation campaign on question generation and boosting research on computational models of question generation.

From this workshop several articles were related directly to the question generation topic that has been presented on this state of the art. As so, we decided to highlight some of the works presented on this workshop.

The first work here considered presents a generic framework for question generation systems (Section 2.3.1) which proposes a generic analysis of different systems and suggests a basic structure to organize any question generation system. The second describes a method for generating questions from queries (Section 2.3.2). For the third question generation is applied to improve quality of questions made by users (Section 2.3.3), highlighting some of the most common errors made by users by analyzing information gathered from several question answering websites. The last paper presents question classification schemes (Section 2.3.4) which constitutes presents a possible organization by categories of different questions.

2.3.1 Framework for question generation

Systems Engineering and Computer Science of Federal University of Rio de Janeiro (Brazil) (Silveira, 2008), proposes a generic structure to organize question generation. To this end, the question generation process should be broken into several proposed modules:

- Pre-processing module - responsible for annotating the text with metadata in order to obtain a structured representation;
- Information extracting module - configures, populates and instantiates a situation model[10];

[9] http://www.questiongeneration.org

[10] "When people experience a story, the phase of comprehension is where people form a mental representation about the text. The mental representation that is formed is called a situation model." (from en.wikipedia.org)

- Formalization and annotation module - encodes the text into a formalism that allows inductive and abductive inference and comparison to a previous built knowledge base;
- Representation expansion module - solves questions that require inference with access to the knowledge base;
- Question design module - applies a group of question models and generates questions. In this module natural language generation techniques are applied;
- Evaluation module - filters low quality questions. Can be automated or human-assisted.

This base structure allows developed question generation systems to be compared more easily, enabling the improvement of specific modules that are under-performing when compared to other developed systems. Some of the modules may not exist on some systems. For example, none of the systems presented above implements the "Representation expansion module" but are extremely focused on the pre-processing and information extracting modules over all.

2.3.2 Automatic question generation from queries

The task of generation of questions from queries (Lin, 2008), proposed by Lin from *Microsoft Research Asia* (China), seeks to find the user goals with the query made, learn templates and normalize queries made by user on search engines.

To this end, several question-answer archives can be studied like Yahoo! Answers[11] or Ask.com[12] and the questions would be paired with queries from logs of query based search engines, for example, MSN/Live Search[13], Yahoo![14] or Ask.com, creating a corpus of relations between queries and questions made by users for training and testing.

The system would then be evaluated by two perspectives:

- intrinsic - how well a method can generate questions given a query;
- extrinsic - how the questions generated can improve the user's search efficiency comparing to the current query suggestion services.

This proposal can be used to find rules for question generation from keyword based topics. If one has a name of something, it can be used to search for information about it, and one could then try to

[11] http://answers.yahoo.com
[12] http://www.ask.com/
[13] http://www.live.com/
[14] http://www.yahoo.com/

transform that name into several questions depending on to what it refers to. Using a part of speech tagged, trained corpus of questions to query relations, one could try to automate the creation of rules for question generation.

For example, a query for "Leonardo da Vinci" could be transformed into the questions "Who was Leonardo da Vinci?" or "When was Leonardo da Vinci born?". Then this relation could be used as template for other people's names. By analyzing both query based search engine logs and question based ones, the relation between the queries and the questions can be discovered by seeking the most frequent questions for a selected keyword.

2.3.3 Generating high quality questions from low quality questions

This work (Ignatova et al., 2008), proposed by *Ubiquitous Knowledge Processing Lab of the Computer Science Department of Technische Universitat Darmstadt* (Germany), proposes the application of question generation to transform low quality questions into higher quality questions, seeking to improve the chance of getting correct answers.

It is based on the analysis of several question and answer platforms like Yahoo! Answers or WikiAnswers[15]. To this end, five main factors of bad quality had been identified:

- Misspelling - example: *Hou to cook pasta?*
- *Internet slang - example: How r plants used 4 medicine?*
- *Ill-formed syntax - example: What Alexander Psuhkin famous for?*
- *Keyword search - example: Drug classification, pharmacodynamics*
- *Ambiguity - example: What is the population of Washington?*

In order to improve these questions some main tasks may be applied:

- Spelling and Grammar Error Correction - Allows the correction of some Misspelling and Ill-formed syntax errors. Internet slang also should be corrected in this task;
- Question generation from a set of keywords - Allows the correction of keyword search and solves ambiguity, but it is only possible if enough context information is provided about the set of keywords.

[15]http://wiki.answers.com

This is specially important since most of the available contents from the Internet, one of the largest data source that can be used for question generation, may contain one or more of these errors that can affect the output of a question generation system.

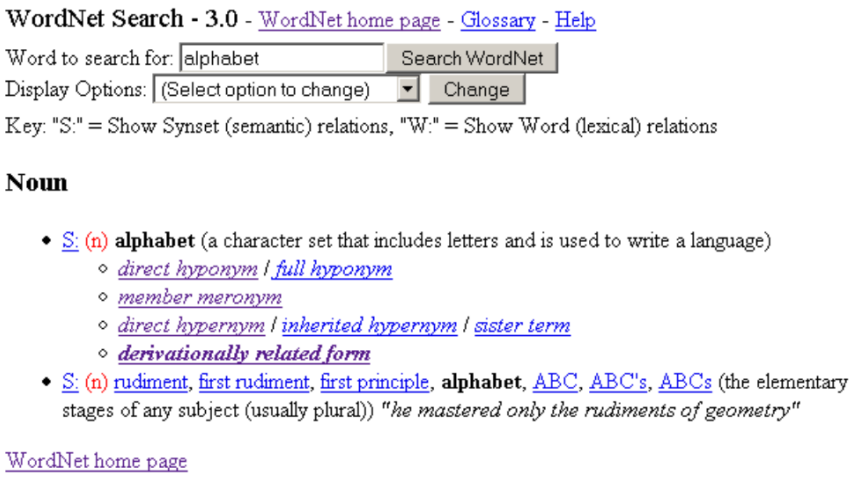
2.3.4 Question Classification Schemes

This work (Graesser et al., 2008) developed at University of Memphis, United States of America, seeks to define question classification schemes. On this article the following question categories were suggested:

1. Verification: questions that invite for yes or no answer
2. Disjunctive: Is X, Y or Z?
3. Concept completion: *who*, *what*, *when* and *where* questions
4. Example: What is an example of X?
5. Feature specification: What are the properties of X?
6. Quantification: How much? How many?
7. Definition: What does X mean?
8. Comparison: How is X similar to Y?
9. Interpretation: What does X mean?
10. Causal antecedent: Why/how did X occur?
11. Causal consequence: What next? What if?
12. Goal orientation: Why did an agent do X?
13. Instrumental/procedural: How did an agent do X?
14. Enablement: What enable X to occur?
15. Judgemental: What do you think of X?

This can be used to organize the patterns for question generation and some of the classifications may require additional mechanisms in order to improve the quality of questions generated. For example, to create comparison questions, the system has to know that there is a characteristic relation between two or more things. In the case of enablement questions, the system has to be able to relate sequences of events that are base to another event. In both situations what is important is the relation, yet they are not of the same type.

Figure 2.2: WordNet - "Alphabet" search results



2.4 Distractors Sources

2.4.1 WordNet

WordNet[16] is a large lexical repository of names, verbs, adjectives and adverbs for English. This project is coordinated by *Cognitive Science Laboratory* of *Princeton University*.

The relations between words are organized by *Synsets*[17] such as:

- hyponym - generic to specific word (example: alphabet - latin alphabet)
- meronym - member of (example: alphabet - letter)
- hypernym - class of (example: alphabet - character set)
- derivationally related form - related words (example: automobile - automobilist)

This lexical database is available on the official WordNet web site, and it can to be used on-line or downloaded. A book version is also available, *WordNet - An Electronic Lexical Database* (Fellbaum, 1998).

This database has been used by CAID as main source for distractors selection and on WebExperimenter as a potential distractor source.

[16] http://wordnet.princeton.edu/ - Visited on December 2009
[17] "Groups of words that are roughly synonymous in a given context" (from wordnet.princeton.edu)

2.4.2 Wikipedia

Wikipedia[18] is a large on-line article repository on several topics developed by the community, featuring several languages with a version to each. The Portuguese version at the moment has over 500 000 articles available and keeps growing each day.

These articles are linked together by keywords that can be explored in order to increase the size of the corpus available for distractor generation. Most articles are also organized by categories. For example, Modern Art and Baroque art articles belongs to the Art Movements category, so while generating a question based on a sentence of a modern art article, distractors could be searched on Baroque art article. This way it is possible to select distractors of higher quality by searching them on texts on the same category.

2.4.3 Dictionaries

Dictionaries are compilations of words, usually of a specific language, alphabetically organized and with definitions and other information. This information source can help relate two or more words when searching for possible distractors.

From the dictionaries available *Wiktionary*[19] is an on-line twin project of Wikipedia. This project features over 70000 words with the respective meanings, etymology, pronunciation and translations. The downside is the lack of a defined structure for each page, that may increase the difficulty of extracting the available information.

2.4.4 FrameNet

The Berkeley FrameNet project[20] coordinated by *International Computer Science Institute* in *Berkeley* (California, United States) is a lexical resource for English that explores, with corpus evidence, each possible sense for each word.

The database developed by this project is based on a semantic frame concept that describes situations (eating, removing, ...), objects or events, presenting relations between lexical units (pair of a word and a meaning). One of the major differences to WordNet is that all the semantic frames are supported by examples that relate all elements it contains. This project can be used for the same purposes of WordNet, but has a different organization structure.

[18] http://www.wikipedia.org
[19] http://www.wiktionary.org
[20] http://framenet.icsi.berkeley.edu - Visited on December 2009

The Berkeley version of the project is described on the ebook *FrameNet II: Extended Theory and Practice* (Ruppenhofer et al., 2006) available on the web site.

There is also a Spanish version of FrameNet[21] under development by *Autonomous University of Barcelona* (Spain) and International Computer Science Institute in *Berkeley* (California, United States) and a German version by *University of Texas* in *Austin* (United States).

[21] http://gemini.uab.es:9080/SFNsite

3 Rule based generation

This chapter presents a system for the multiple choice test generation supported by rules created manually. First the system architecture is described on 3.1, followed by the question generation on 3.2, the distractor generation module on 3.3 and finally the edition of question items on 3.4.

In the end of this chapter two evaluations of the system with a different set of rules are presented on 3.5.

3.1 System Overview

This multiple choice test generation system (Figure 3.1) starts by analyzing the selected corpus through a Natural Language Processing Chain (from now on NLP chain) developed at L2F (Hagège et al., 2008). On this stage, dependencies and information about the recognized words are added to the corpus, producing a part of speech tagged corpus with some dependencies identified.

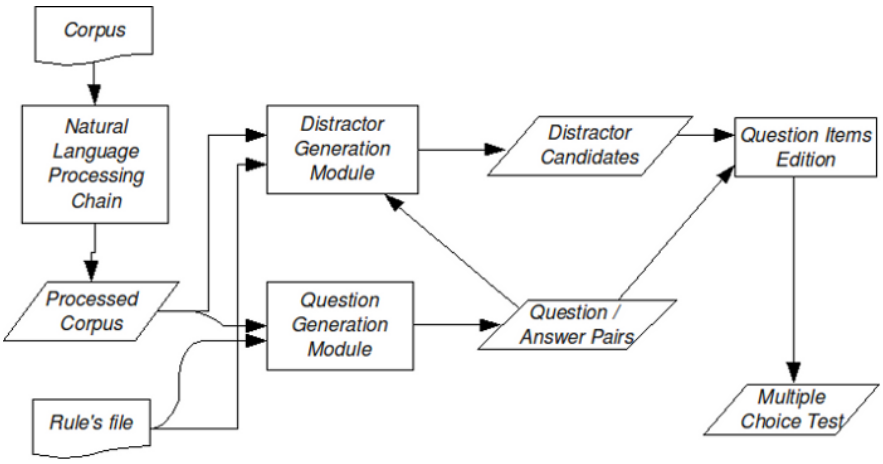
The next input component is a file with rules that specify, for each dependency type that is detected by the NLP chain, what questions should be generated. Currently this set of rules is manually developed.

The processed corpus and the rule's file are then received by the question generation module (described on Section 3.2), which will match the rules with dependencies found in the processed corpus and create a question item. Each item is composed by a question, the correct answer and the answer features identified by the NLP chain (for example, the answer "cantor" is identified by features noun, common, masculine and singular).

This information is then passed to a distractor generation module that will seek distractors on a distractor selection source. At the moment the only distractor selection source is the processed corpus it self (Section 3.3.1).

In the last stage, the output of the question generation and distractor generation modules is given to the user for approval or to discard the question items. For the approved question items, the user can edit them if needed.

Figure 3.1: Rule based architecture overview



3.2 Question Generation Module

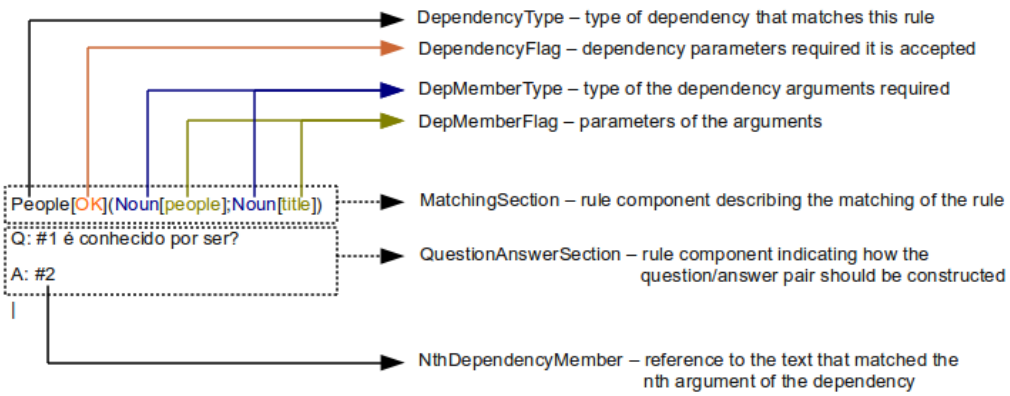
This module is responsible for generating question/answer pairs, transforming the corpus processed by the NLP chain. In the next pages, the syntax of the rule's file is described (Section 3.2.1) and a full example of the question generation process is presented (Section 3.2.2).

3.2.1 Rules

The rule's file is one of the main inputs of the question generation system. These files can be described by the following Backus-Naur Form (BNF):

```
<rulefile> ::= <rule>+
<rule> ::= <matchingSection>("\n"<questionAnswerSection>)+"\n|"
<questionAnswerSection> ::= "Q: "<questionText>"\n"
                            "A: "<answerText>"\n"
<matchingSection> ::= <dependency>"("<depMember> (","<depMember>")*")"
<dependency> ::= <dependencyType> |
                 <dependencyType>"["<depFlag>(","<depFlag>)*"]"
<depMember>  ::= <depMemberType> |
                 <depMemberType>"("<depMemberFlag>
                                  (","<depMemberFlag>)*")"
<questionText> ::= <text> (<#nthDependencyMember> <text>)+ "?"
<answerText>   ::= <#nthDependencyMember>
<nthDependencyMember> ::= "#"(0-9)+
```

Figure 3.2: Rule example



The dependency type, dependency member type, dependency flag and dependency member flag are defined by the processing of the corpus by the NLP Chain. This can be described by the example on Figure 3.2.

This rule is activated if we find a dependency of the type "People" with a flag "OK", on the corpus processed by the NLP chain, with two members ("Noun[people]" and "Noun[title]") of the type "Noun", one with the flag "People" and the other with "Title". When this rule is matched, #1 and #2 are replaced by the words that matched "Noun[People]" and "Noun[title]" respectively.

In order increase the rule's flexibility, dependency type can also be replaced by a wild-card "*" which means that any type is accepted as long as the dependency flags are assured. For example:

```
Business[profession](*[people];*[profession])
Q: Qual é a profissão de #1?
A: #2
|
```

In this case, the system seeks dependencies of the type "Business" with a flag "profession", whose two members are of any type and flagged as "people" and "profession" respectively.

3.2.2 Example

The question generation module, for each of the corpus units, seeks to match the loaded rules with the dependencies identified by the NLP Chain. Each time a match is found a question/answer pair is generated, with detailed information about the answer that is needed for the distractor generation step.

We will describe this operation with the following example:

1. The question generation module loads the following rule:

```
People[OK](Noun[people];Noun[title])
Q: #1 é conhecido por ser?
A: #2
|
```

2. Then, while searching the processed corpus, the previous rule matches the following dependency:

```
<NODE num="44" tag="NOUN" start="23" end="23">
<FEATURE attribute="TITLE" value="+"/>
...
</NODE>

...
<NODE num="54" tag="NOUN" start="24" end="25">
<FEATURE attribute="PEOPLE" value="+"/>
...
</NODE>

...
<DEPENDENCY name="PEOPLE">
<FEATURE attribute="OK" value="+"/>
<PARAMETER ind="0" num="54" word="Michael Jackson"/>
<PARAMETER ind="1" num="44" word="cantor"/>
</DEPENDENCY>
```

3. The question generation module would then generate a question/answer pair by replacing the #1 and #2 with the words “Michael Jackson” and “cantor”, generating the following pair:

```
Michael Jackson é conhecido por ser?
cantor
<answerNodes>
lunitNumber=1194;
<node>
arg1Text=cantor;arg1Type=NOUN;arg1Option=COMMON;
arg1Option=TITLE;arg1Option=MASC;arg1Option=SG;
arg1Option=NOUN;arg1Option=HMMSELECTION;
arg1Option=LAST;
```

```
</node>
</answerNodes>
```

Following the question generated, the module outputs all information available about the answer (between "answerNodes" XML tags). That information is the base of the distractor generation.

3.3 Distractor Generation Module

The distractor generation module receives the output of the question generation module and then applies a distractor generation technique. The development of this module can be based on one or more sources in order to achieve higher quality distractors.

At the moment the only module supported by this system is based on the shared features search described next, but in the future other sources can be explored (for example WordNet).

3.3.1 Distractor Search

The distractor search consists in locating in the corpus words that share traces with the answer to the question. This simple method allows the generation of distractor candidates.

For each question/answer pair, the module searches the processed corpus for words with the same traces of the answer. Then, from that list, it selects a group of distinct words to be proposed as distractors. Considering the example question/answer pair presented in Section 3.2.2, the distractor search would seek the processed corpus for words that share the type "Noun", as well as all the options listed inside the node XML tag, returning the following distractor candidates:

```
<distractors>
arquitecto
segurança
embaixador
</distractors>
```

At the moment, the search is made by giving priority to matches near the correct answer found in the processed corpus. This is done in order to increase the chance of getting distractors that are topic related with the correct answer. The number of distractors to be located and the threshold of characteristics that the candidate distractor needs to share with the correct answer are both variables on the distractor generation process. The threshold can be a value between 0% and 100%. This controls the traces that distractors need to share with the answer to be considered candidates.

For the current prototype the threshold is set to 100% in order to improve the quality of the distractors generated, at the cost of not being able to find enough distractors. In future work, a study on which features should be prioritized can allow this threshold to be lowered while assuring the required quality. This change would lead to the generation of more distractors for the same corpus.

3.4 Question Items Edition

The question/answer pairs and the distractor candidates are presented to the user, who will accept, discard or edit the generated question item.

Currently all the question item edition is done manually.

3.5 Evaluation

This system will be evaluated by using two different sets of rules. The first evaluation uses a large corpus and the second uses the same corpus and in addition, an article from wikipedia.

Both evaluations start by presenting the Corpora and Evaluation Parameters, describing the corpus and the set of rules, followed by an evaluation of the Question/Answer pairs generated and ends with an analysis of the distractors generated.

3.5.1 First Evaluation

3.5.1.1 Corpora and Evaluation Parameters

For this evaluation six days of newspaper "Público" from 01/01/1994 to 06/01/1994 were used. The selected corpus contains news of different types and subjects.

For the question generation, five rules were created based on dependencies about people, location, nationality and culture. The rule's file is the following:

```
Culture[OK](Noun[proper];Noun[people];VCPART)
Q: Quem #3 #1?
A: #2
|
People[OK](Noun[people];Noun[title])
Q: #1 é conhecido por ser?
A: #2
```

```
|
Nationality[OK](Noun[people];Adj[nationality])
Q: Qual é a nacionalidade de #1?
A: #2
|
Location[OK](Noun[city];Noun[proper])
Q: #1 pertence a?
A: #2
|
Location[OK](Noun[capital];Noun[country])
Q: #1 é a capital de?
A: #2
|
```

The distractor generation was based on word alike search, seeking to generated three distractors per question, that matched all the options of the correct answer.

3.5.1.2 Question/Answer Pairs Generation

The first step on the evaluation of the question/answer pairs generated was simply to accept or discard each one of them. For example, consider the question "Qual é a nacionalidade de Disney?", "Disney" can refer to "Walter Elias Disney" or "Roy O. Disney". Due to this ambiguity, this question should be discarded.

Moreover, the model of test item review of CAID (see 2.2.1) was then used on the items generated by the prototype. They were evaluated by the degree of review needed in order to be used on a multiple choice test, which was classified as:

- None/Minor - test item requires no corrections or only minimal corrections (punctuation, article introduction or spelling corrections), for example:

```
Qual é a nacionalidade de Michael Jackson?
norte-americano
```

- Moderate - test item requires the removal or reordering of words, for example "el-" has to be removed on the answer:

```
D. Sebastiao é conhecido por ser?
el-rei
```

- Major - test item requires grammatical correction, for example the question has to be changed since "El País" refers to a newspaper and not a person:

```
Qual é a nacionalidade de El País?
espanhol
```

The system generated 321 question/answer pairs. From this 249 were accepted (77.5%). The set of pairs accepted was then divided as follows:

- 203 requiring minor review (81.5%);
- 40 moderate review (16.1%);
- 6 major review (2.4%).

3.5.1.3 Distractor Generation

The test items that were accepted during the last step, were evaluated by degree of review their distractors needed per question/answer pair. The classification applied to each set of distractors was:

- None/Minor - up to one distractor needs minor corrections (for example, gender agreement), replacement or is missing. For example:

```
Clark Terry é conhecido por ser?
cantor
<distractors>
pintor
ponto
actor
</distractors>
```

- Moderate - all the test item distractors require some corrections and up to one distractor needs replacement or is missing. For example the first and last distractors needs to be replaced by the nationality they refer to ("francesa" and "brasileira" respectively):

```
Qual é a nacionalidade de Rudy Walem?
belga
<distractors>
parisiense
```

```
etíope
carioca
</distractors>
```

- Major - two or more distractors generated are missing or have to be replaced. For example not enough distractors were generated on the following test item:

```
Mel Lewis é conhecido por ser?
baterista
<distractors>
presidente
</distractors>
```

With the same corpus 81 (32.5%), 31 (12.5%) and 137 (55.0%) sets of distractors required minor, moderate and major revisions, respectively.

One of the major problems on shared features search, is that it does not take in to consideration relations between words when it seeks for distractors. So sometimes it selects words as candidate distractor that are more generic or more specific than the answer. For example, consider a question item about the nationality of a person and the correct answer "Portuguese", shared features search selects "European" as distractor. Both words share the same POS tags yet one is more specific then the other. This could be solved by using the direct hypernym relations present on WordNet.

3.5.2 Second Evaluation

3.5.2.1 Corpora and Evaluation Parameters

For this evaluation, the same six days of newspaper "Público" from 01/01/1994 to 06/01/1994 were used. Additionally, the wikipedia article "História de Portugal" was used to analyze the question generation component on a smaller and different type corpus.

For the question generation, 26 rules were created based on dependencies created by (Santos, 2010). The rule's file is available on Appendix A.

The distractor generation was based on shared features search, seeking to generated three distractors per question, that matched all the options of the correct answer. This component was not evaluated for the wikipedia article since it is not large enough for proper distractor generation with the current parameters.

3.5.2.2 Question/Answer Pairs Generation

The same metrics presented on the first evaluation on 3.5.1.2 were used for this evaluation.

With the newspaper corpus the system generated 316 question/answer pairs. From this 215 were accepted (68.0%). The set of pairs accepted can be divided as 204 requiring minor (94.9%), 9 moderate (4.2%) and 2 major (0.9%) revisions.

An example of a question/answer pair that requires minor revision, where the preposition "de" and the article "os" have to be contracted into "dos":

```
Daniel Kenedy Pimentel Mateus de os Santos reside em?
Bissau
```

As an example of a moderate revision pair we have a pair that requires the removal of the reference to the second person on the question, keeping only "Mel Lewis":

```
Mel Lewis e o cantor Clark Terry é conhecido por ser?
baterista
```

In the case of the major revision pair we have the following example where the question should be changed to plural since it refers two persons:

```
Hall e Stillman é conhecido por ser?
investigadores
```

Finally an example for the discarded pair:

```
Maria é conhecido por ser?
actrizes
```

In the case of the wikipedia article "História de Portugal" 13 question/answer pairs were generated, 12 (92.3%) were accepted. All the pairs accepted only required minor revisions and are the following:

```
Afonso Henriques reside em?
coimbra
```

```
Afonso I é conhecido por ser?
```

```
rei

Inocêncio II é conhecido por ser?
papa

D. Sebastião é conhecido por ser?
rei

Coroa Portuguesa fundou?
conselho ultramarino

D. João VI é conhecido por ser?
rei

D. Carlos é conhecido por ser?
rei

Augusto é conhecido por ser?
imperador

Vespasiano é conhecido por ser?
imperador

Diocleciano é conhecido por ser?
imperador

Junot é conhecido por ser?
general

Loison é conhecido por ser?
general
```

The pairs discarded were:

```
Católicos é conhecido por ser?
```

```
Reis
```

In the history themed corpora the questions generated should be in the past tense instead of the present. Exploring the corrections required for each theme would be an interesting point of improvement for this system in the future.

3.5.2.3 Distractor Generation

Only the generation based on the newspaper is going to be analyzed regarding distractor generation, since the system did not found enough distractors in the wikipedia article. Taking in to account the question/answer pairs accepted on the previous step of the evaluation for the newspaper corpus, 106 required minor (49.3%), 34 moderate (15.8%) and 75 major (34.9%).

An example of a test item that required none/minor revision, that in this case requires gender correction on “conhecido” to “conhecida”, is:

```
Rosemary Clooney é conhecido por ser?
vocalista
<distractors>
trompetista
violoncelista
baterista
</distractors>
```

In the case of the moderate revision, we have the following example where a distractor is missing:

```
Pablo Chisonne é conhecido por ser?
juiz
<distractors>
Presidente
Papa
</distractors>
```

Finally, an example of a test item that requires major revision due to the lack of distractors:

```
Jian Wang é conhecido por ser?
violoncelista
<distractors>
</distractors>
```

3.5.3 Evaluation Results Summary

The results of the evaluations for this system are summarized on the following table:

Table 3.1: Rule Based Generation - Evaluation results

	First Evaluation	**Second Evaluation**
Items Generated	321	316
Accepted	249 (77.5%)	215 (68.0%)

Question/Answer Pair Generation		
None/Minor	203 (81.5%)	204 (94.9%)
Moderate	40 (16.1%)	9 (4.2%)
Major	6 (2.4%)	2 (0.9%)

Distractor Generation		
None/Minor	81 (32.5%)	106 (49.3%)
Moderate	31 (12.5%)	34 (15.8%)
Major	137 (55.0%)	75 (34.9%)

4 Pattern Bootstrap Generation

4.1 System Overview

Based on the pattern discovery presented by (Silva, 2009), a question generation system was created. This system main components are the following (Figure 4.1):

- Seed creation described in Section 4.2;
- Pattern bootstrap described in Section 4.3;
- Question/Answer generation described in Section 4.5;
- Distractor generation described in Section 4.6;
- User interface described in Section 4.7.

Given several question/answer pairs, the system starts by grouping them into sets of two, denominated seeds. In each seed, the first question/answer pair is used to discover patterns and the second is used to validate the patterns. These are then used to identify text segments that match the pattern information, which are transformed by the system to generate questions with answers. Finally, distractors are found for each one of the question/answer pair generated.

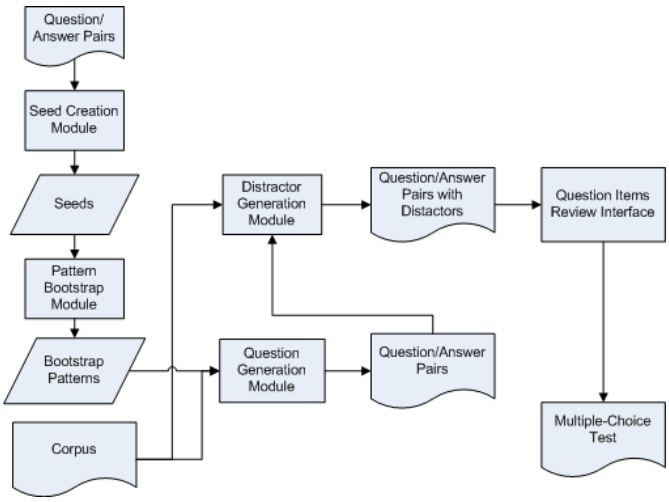
In what regards the syntactic analysis of questions, we use the Berkeley Parser (Petrov & Klein, 2007) trained on the QuestionBank (Judge et al., 2006), a treebank of 4,000 parse-annotated questions.

In what concerns the question category, we use Li and Roth's two-layer taxonomy (Li & Roth, 2002), which is one of the most widely known taxonomies for question classification, consisting of a set of six coarse-grained categories (*ABBREVIATION*, *ENTITY*, *DESCRIPTION*, *HUMAN*, *LOCATION* and *NUMERIC*) and fifty fined-grained ones.

4.2 Seed Creation

One of the base components of the system described on this chapter is the seed. Each seed is composed by two question/answer pairs. The first question/answer, preceded by "QA:", is used to extract patterns

Figure 4.1: Pattern Generation - System overview



with the selected search engine; the second component, preceded by "VQA:", is used to validate the discovered patterns.

Example 15: Example seed

QA:How many years did Rip Van Winkle sleep?;twenty VQA:How many years did Sleeping Beauty sleep?;100

■

For this process to work, both questions need to have the same syntactic structure and refer to similar concepts. The previous example is a high quality seed and the next one is a low quality seed.

Example 16: Seed with similar structure only

QA:Who pulled the thorn from the Lion's paw?;androcles VQA:What was the color of Christ's hair in St John's vision?;white ■

In order to assure the quality of these seeds, the system sorts the given question/answer pairs by syntactic structure, presented on the previous example. Then the pairs are grouped by first question component, so they both ask the same type of question.

Example 17: Seeds with similar structure and first component content

Correct question pairing: QA:What color was the Maltese Falcon?;black VQA:What color was Moby Dick?;white

Incorrect question pairing: QA:What was the language of Nineteen Eighty Four?;newspeak VQA:What was the color of Christ's hair in St John's vision?;white ■

Finally, using a pre-trained question classifier analyzer, the questions are grouped by expected answer category.

Algorithm 1 Question rewrite patterns bootstrapping algorithm

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

Table 4.1: Components identified by the parser for "How many years did Rip Van Winkle sleep?"

Label	WHNP	VBD	NP	VB
Content	how many years	did	rip van winkle	sleep

Example 18: Seeds with question category restriction, same structure and first component

QA:How many years did Rip Van Winkle sleep?;twenty VQA:How many years did Sleeping Beauty sleep?;100

QA:What is the last word of the Bible?;amen VQA:What is the Hebrew word for peace used as both a greeting and a farewell?;shalom

QA:Who was Don Quixote's imaginary love?;dulcinea VQA:Who was Shakespeare's fairy king?;oberon ■

4.3 Pattern Bootstrap

The pattern discovery, based on the work developed by (Silva, 2009) (Algorithm 1), is composed of several steps that are going to be described using the following seed:

Example 19: Example seed

QA:How many years did Rip Van Winkle sleep?;twenty VQA:How many years did Sleeping Beauty sleep?;100

■

On this seed, the question preceded by "QA:" is used to search for patterns and the one preceded by "VQA:" is used to verify and test the precision of the generated patterns.

The first step on the pattern discovery is to analyze both questions, with a parser, identifying the components. Using the example seed, the first question would have the components presented on the Table 4.1, where "Labels" and "Content" represent the POS tags and the words assigned to the each component, respectively.

Then, permutations of the first question/answer pair (except the first component of the question) and a wild-card are created and sent to the selected search engine. Using the components identified on Table 4.1 and the answer "twenty" the following permutations are created for the example seed:

Example 20: Search query creation

[[did, rip van winkle, sleep, *, twenty], [rip van winkle, did, sleep, *, twenty], ...] ■

Permutations that start or end with the wild-card are removed from this set because the wild-card would match all the query result text before or after the components permutation, respectively, creating patterns that would be too specific for that text.

The system then analyzes the query results locating the permutation components for each of the results. With this information an answer pattern is generated that contains the question POS tags, the text that matched the wild-card and an answer component.

Each of generated patterns are then tested by replacing the question POS tags with the second question equivalent contents and are sent to the search engine. The percentage of query results elements that contain this text sequence is calculated, giving the pattern a score. If this score is over a certain percentage, the pattern is registered as a **strong pattern**.

If no patterns were registered for a given seed, the main verb of the question is replaced by each of its inflections and the auxiliary verb is removed on the sequences to send to the search engine. Then the validation process selects which patterns should be registered, by applying the same modification to the main verb of the verification question. Each pattern registered by this process is called **inflected pattern**. This allows the system to find patterns that are not in the same verbal tense of the question.

Example 21: Inflected search query

[[rip van winkle, slept, *, twenty], ...] ■

If no patterns are registered using the previous described processes, the process is repeated but the query used to generate the pattern contains only the noun components of the question, the answer and a wild-card. This simplified generation creates what we call **weak patterns**.

Example 22: Weak search query

[[rip van winkle, *, twenty], [twenty, *, rip van winkle]] ■

With the example seed presented the system generated the following inflected pattern:

Example 23: Example seed bootstrap results

Question category: Numeric Count

Question tags: WHNP VBD NP VB

Inflected Pattern: NP[QUESTION,2,0] VBD[QUESTION,3,0] IN![for] CD[ANSWER,0]

Answer POS: CD

Score: 0.625 ■

Each component of the pattern is composed by a POS tag, followed by brackets containing the identification of the question/answer pair component (structured has "*type*, *component number*, *sub-component number*"). Pattern components that have a "!" at the end of the POS tag, the brackets contain text that should found on the corpus in order to activate the pattern, the text matched is not used on the generation of the question/answer pair.

4.4 Corpus Sources

This system can currently use either a text file given by the user, an article from wikipedia or a generic webpage. In what concerns the generic webpage and wikipedia articles, the system retrieves the latest version of the respective webpage and removes html and scripting code from it. Additionally, in the wikipedia article tables, images and footnote references are removed in order to improve the page parsing results.

This corpus source interface can be easily extended to new sources in the future.

4.5 Question/Answer Pair Generation

For the question/answer pair generation phase, using a set of patterns discovered on the previous step and a corpus, the patterns are matched against the corpus (Algorithm 2) and a group of filtering rules are applied.

For a pattern to match, the corpus text segment is required to have the same parse tree structure of the pattern. When a match occurs, the matching segment is analyzed on the following aspects:

1. Answer category match;
2. Query reconstruction test;
3. Removal of anaphoric references.

On the answer category match, each word of the answer is tested to see if it matches the expected category of the question. The item is discarded if this verification fails.

Example 24: Answer category match

(Human:Individual) Who was François Rabelais? An important 16th century writer. ■

Then, for strong and inflected patterns the query is reconstructed and sent to the selected search

Algorithm 2 Bootstrap pattern matching algorithm

```
procedure FIND-SEQUENCES(sequence : TreeMatcherSequence, previousSequencesInProgress :
List<TreeMatcherSequence>, tree : SyntacticTree)
    sequencesInProgress ← []
    for each sequence in previousSequencesInProgress do
        testSequence ← SEQUENCE-MATCH-TREE(sequence, tree)
        if SEQUENCE-IS-MATCHED(testSequence) then
            matchedSequences ← ADD(matchedSequences, testSequence)
        else
            if IS-TREE-NOT-ROOT(tree) and IS-SEQUENCE-IN-PROGRESS(testSequence) then
                matchedSequences ← ADD(matchedSequences, testSequence)
            end if
        end if
    end for
    for each child in tree.children do
        if IS-TREE-NOT-TERMINAL(child) then
            subResults ← FIND-SEQUENCES(sequence, sequencesInProgress, child)
            for each subResult in subResults do
                if IS-SEQUENCE-MATCHED(subResult) then
                    matchedSequences ← ADD(matchedSequences, subResult)
                else
                    if IS-SEQUENCE-IN-PROGRESS(subResult) then
                        sequencesInProgress ← ADD(sequencesInProgress, subResult)
                    end if
                end if
            end for
        end if
    end for
    if IS-TREE-NOT-ROOT(tree)( )then
        matchedSequences ← ADD(matchedSequences, sequencesInProgress)
    end if
    return matchedSequences
end procedure
```

engine and if a minimum amount of results is returned, the question and answer are formulated. If the triggered pattern is typed as weak, the generation is delegated to the strong or inflected patterns that share both question structure and category. The information of the strong or inflected patterns is used during the query reconstruction to fill in any missing elements that are not supported by the weak pattern.

The last step is to discard any question/answer items that are based on anaphoric references, since this references are not processed by the current system.

Example 25: Anaphoric reference question/answer pair

(Location:City) Where is it? Paris ■

Question/answer pairs that pass all filtering described above are sent to the distractor generation component. Algorithm 3 shows how the question/answer pair generation and distractor generation are integrated to generate question items.

Algorithm 3 Bootstrap based question item generation algorithm

procedure QUESTION-ITEM-GENERATION(*corpus* : text source, *patterns* : patterns bootstraped, *seed* : question-answer pairs used on bootstrap)
  question-items ← []
  question-answer-pairs ← []
  distractor-candidates ← []
  for each *sentence* **in** *corpus* **do**
    parsed-sentence ← PARSE(*sentence*)
    for each *pattern* **in** *patterns* **do**
      results ← GENERATE-QUESTION-ITEMS(*pattern*, *parsed-sentence*)
      question-answer-pairs ← ADD(*question-answer-pairs*, *results*)
    end for
    results ← GENERATE-DISTRACTORS(*patterns*, *parsed-sentence*)
    distractor-candidates ← ADD(*distractor-candidates*, *results*)
  end for
  for each *question-answer-pair* **in** *question-answer-pairs* **do**
    question-item ← SELECT-DISTRACTORS(*distractor-candidates*, *question-answer-pair*)
    question-items ← ADD(*question-items*, *question-item*)
  end for
  return *question-items*
end procedure

4.6 Distractor Generation

As the text is processed during multiple-choice question item phase, the text is analyzed by a named entity recognizer, and components of the text that share the same classification with the question patterns expected answer category are stored.

This index is then used to select distractor candidates for the generated questions, giving preference to entities closer to the correct answer. This preference is used in order to find candidates that share context with the generated question/answer pair.

4.7 Interfaces

The system has a web-based interface. This interface is controlled by a Java Servlet that makes requests to a set of web-services.

A pattern generation interface is presented in Section 4.7.1, followed by a multiple-choice test generation interface in Section 4.7.2 and then the available web-services are described in Section 4.7.3.

4.7.1 Pattern Generation Interface

The pattern generation interface requests two question/answer pairs (Figure 4.2), which are validated by the system to ensure that they have the same syntactic structure.

Figure 4.2: Pattern Generation - Seed request

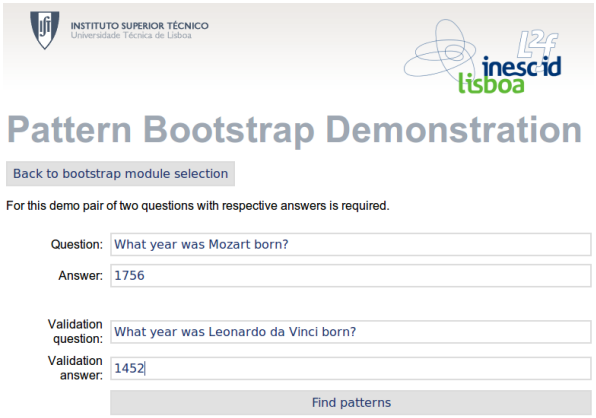
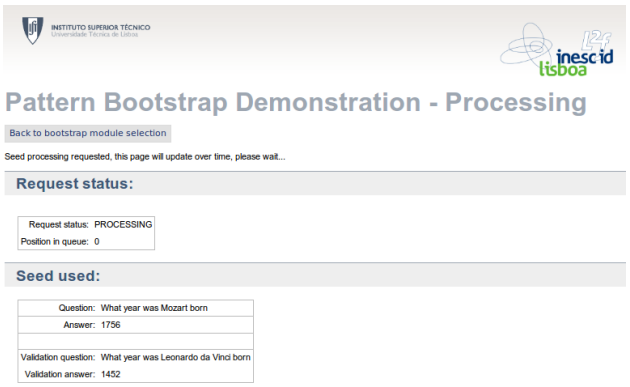


Figure 4.3: Pattern Generation - Processing



This information is then sent by the servlet to the web-service responsible by the pattern bootstrap and the user is notified that is request is being processed (Figure 4.3).

When the processing of the given seed is done, the web-interface presents the results to the user (Figure 4.4).

The user can then return to the start of the application and input another seed for processing.

4.7.2 Multiple-choice Test Generation Interface

The multiple-choice test generation interface asks the user for a wikipedia article name, number of questions and distractors per question to be generated (Figure 4.5).

Using the information requested, provided the article is available on wikipedia, the servlet places a processing request in web-service and waits for it to complete processing. While waiting for the web-

Figure 4.4: Pattern Generation - Results

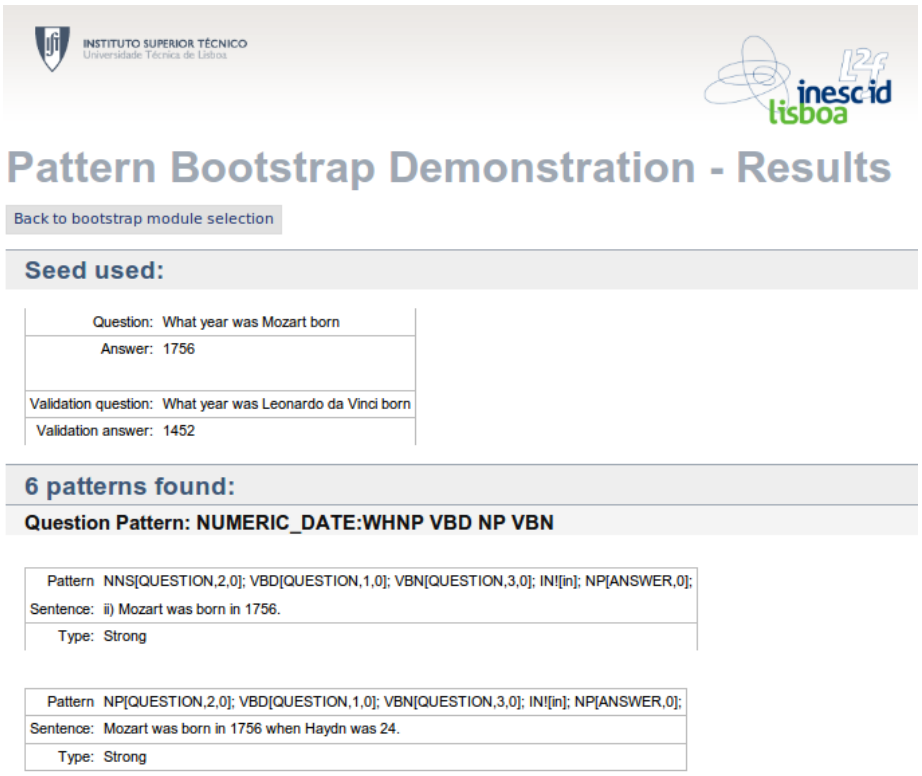


Figure 4.5: Multiple-choice test Generation - Request

Figure 4.6: Multiple-choice test Generation - Processing

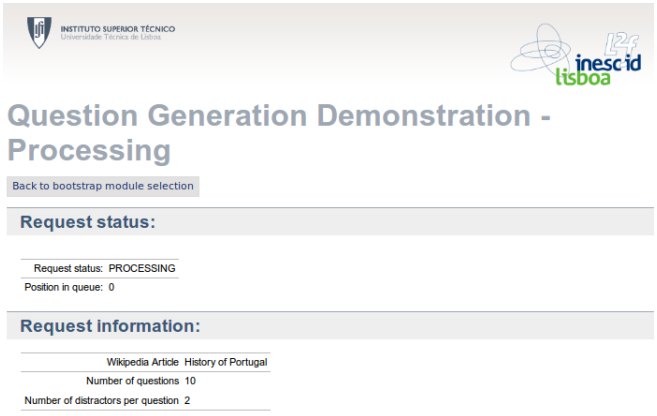
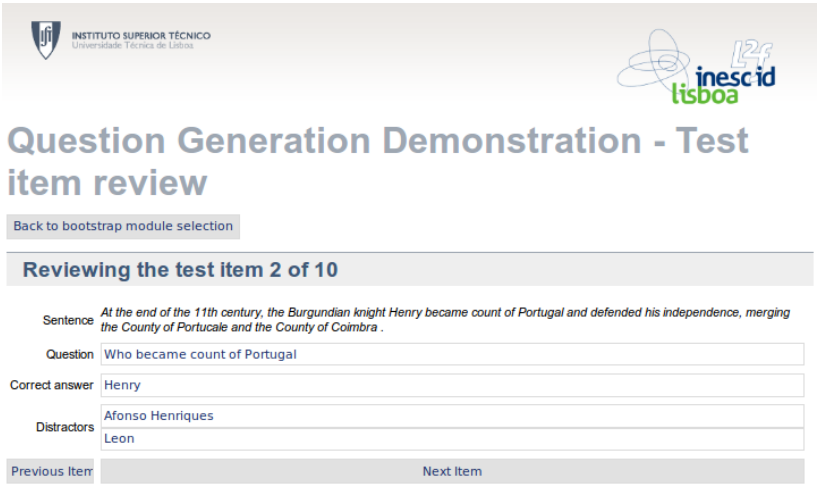


Figure 4.7: Multiple-choice test Generation - Item review



service to finish, the request information is presented to the user (Figure 4.6).

When the processing is finished the web-interface presents a review interface for each item generated, presenting the question, answer, distractors and sentence that originated the test item. The user can edit all the fields for each test item (Figure 4.7).

When all the review is done the web-interface presents a complete multiple-choice exam and the parameters used to generate it (Figure 4.8).

Figure 4.8: Multiple-choice test Generation - Results

4.7.3 Web-service Interface

Two web-services were created to allow the access of the demonstration servlet to the bootstrap and test item applications. These web-services can be accessed by any other application allowing the use of the system independently of the platform.

The pattern bootstrap web-service has the following methods available:

- postRequest(String ipAddress, String question, String answer, String validationQuestion, String validationAnswer) - places a request on the pattern bootstrap system to create patterns based on the given seed;
- getRequestStatus(int sessionId, String question, String answer, String validationQuestion, String validationAnswer) - retrieves the status of a previous given request;
- getRequest(int sessionId, String question, String answer, String validationQuestion, String validationAnswer) - retrieves the request generation results.

The multiple-choice test web-service has the following methods available:

- postWikipediaRequest(String ipAddress, String articleName, int numberOfDistractors) - places a request on the multiple-choice test generation system to create a test based on the given article and a certain number of distractors per question item;

- getRequestStatus(SessionKey identifier) - retrieves the status of a previous given request;
- getRequest(SessionKey identifier) - retrieves the request generation results.

4.8 Evaluation

For this system evaluation the ability to generate question items with minimal user intervention will be analyzed. First the evaluation parameters are presented in Section 4.8.1, followed by question/answer pairs generation evaluation in Section 4.8.2 and, finally, distractor generation in Section 4.8.3.

4.8.1 Evaluation Parameters

For this evaluation 27 question/answer pairs (in groups of 3 questions per topic) were used to generate a set of 72 seeds for the pattern bootstrap phase. The question/answer pairs used where the following (the format used by the system is *answer ":" question*):

1881 : When was Pablo Picasso born? 1853 : When was Van Gogh born? 1904 : When was Salvador Dali born?

Malaga : Where was Pablo Picasso born? Zundert : Where was Van Gogh born? Figueres : Where was Salvador Dali born?

1973 : When did Pablo Picasso die? 1890 : When did Van Gogh die? 1989 : When did Salvador Dali die?

Pablo Picasso : Who painted Guernica? Salvador Dali : Who painted The Persistence of Memory? Van Gogh : Who painted The Starry Night?

1896 : When did Pablo Picasso start painting? 1909 : When did Salvador Dali start painting? 1881 : When did Van Gogh start painting?

cubism : What is Pablo Picasso's painting style? surrealism : What is Salvador Dali's painting style? impressionism : What is Van Gogh's painting style?

629 : How many paintings did Pablo Picasso do? 864 : How many paintings did Van Gogh do? over 1500 : How many paintings did Salvador Dali do?

Reina Sofia museum : Where is Guernica located? Museum of Modern Art : Where is The Persistence of Memory located? National Galery : Where is The Starry Night located?

1937 : When did Pablo Picasso paint the painting Guernica? 1931 : When did Salvador Dali paint the painting The Persistence of Memory? 1889 : When did Van Gogh paint the painting The Starry Night?

The pattern bootstrap was applied to the seeds and generated 59 patterns. From this, 40 are strong patterns and 19 are inflected patterns. Question category wise *LOCATION_OTHER* has 19 patterns, *NUMERIC_DATE* has 9 patterns and *HUMAN_INDIVIDUAL* has 31 patterns. Weak patterns were not generated since the goal of this evaluation is to generate question items that require little user intervention.

To keep the same context of the questions the wikipedia articles "Renaissance" and "Leonardo" were used as corpus sources. Tables, footnotes and images were removed from these articles to improve the speed and quality of the parsing.

The distractor generation was based on the corpus based distractor source presented in Section 4.6 and the system was requested to generate three distractors per test item.

4.8.2 Question/Answer Pairs Generation

From the two wikipedia articles, the system analyzed 346 sentences and successfully generated 40 unique question/answer pairs. From this set 35 and 5 were supported by strong and inflected patterns, respectively.

From this set 9 question/answer could be used with little or no corrections, 16 could be corrected by choosing another answer suggested by the system or using the information presented on the sentence that originated it and 15 should be discarded.

Most of the questions generated belonged to the category *HUMAN_INDIVIDUAL* which was expected since the pattern bootstrap created more patterns for this category.

An example of a good question/answer pair, extracted from the sentence "The first building to demonstrate this is claimed to be the church of St. Andrew built by Alberti in Mantua.", is the following:

Example 26: Good question/answer pair generated by the bootstrap system
Who built the church of St. Andrew?

- Alberti

■

As example of a question/answer pair that can be corrected using the information on the sentence "In 1502 Leonardo entered the service of Cesare Borgia, the son of Pope Alexander VI ..." we have:

Example 27: Ambiguous question/answer pair generated by the bootstrap system
Who entered the service?

- Leonardo

■

In the last example, the user would have to complete the expression "the service" to "the service of Cesare Borgia" creating a good question/answer pair.

Finally, we have obtained question/answer pairs that should be discarded. For example, from the sentence "Political philosophers such as Niccolò Machiavelli and Thomas More revived the ideas of Greek and Roman thinkers, and applied them in critiques of contemporary government." the system generated the following:

Example 28: Discarded question/answer pair generated by the bootstrap system
Who revived the ideas?

- Thomas Moreless

■

In most situations where the question/answer pair was discarded, the user would have to read more information on the corpus in order to create a proper question. An anaphoric reference resolution component could transform some of this items in to higher quality ones.

4.8.3 Distractor Generation

On distractor generation, taking in account the question/answer pairs that were accepted on the previous phase, 15 had distractors that were ready to be used, 4 required up to one distractor modified and 6 questions had distractors that required complete replacement.

An example of a good question item is:

Example 29: Good test item generated by the bootstrap system
Who built the church of St. Andrew?

- Alberti
- Raphael
- Leonardo da Vinci
- Filippo Brunelleschi

■

The main problems presented on the distractor selection was either the lack of words that would fit the question/answer pair category, so the system could not generate enough distractors, or situations were the name of a painting would be confused with the name of a person, creating a low quality distractor that could be identified in the presence of other test items.

5 Conclusion

Multiple choice test generation is still a topic being analyzed by different approaches and with different goals. Applications range from assistance on the development of multiple-choice tests to the creation of tools to assist the learning of a language.

Some of the presented systems interact with the user in order to improve the quality of the question generation and to allow the user some control over the process (for example, Web Application).

The question generation and the distractor generation tasks can be approached by different technics and use different information sources to achieve their goal. This thesis created two multiple-choice generation systems, one based on hand-crafted rules and another based on automated pattern discovery.

An interface for the automated system was developed, presenting the main components of this system: pattern generation and question item generation. Results for both systems show the ability to generate a several question items that required only minor revisions by the user.

5.1 Contributions

The main contributions of this thesis are the following:

- Creation of dependency patterns for the L2F NLP chain;
- Creation of a rule-based system that generates multiple-choice test items for Portuguese language using the dependencies identified by the NLP chain;
- Development of a bootstrap based system to generate multiple-choice tests, for English language, based on the pattern bootstrap component introduced by (Silva, 2009), primarily used within a QA system. In particular, changes were made to the patterns, storing additional information required for question generation;
- Creation of a interface for the pattern bootstrap generation components - pattern generation and multiple-choice test item generation;
- Creation of a corpus interface that can access and retrieve a clean version of a given website or wikipedia article (removing scripting, tables, images and footnotes from the text);

- Creation of web-services to allow other systems to be developed based on this bootstrap system.
- Corrections on the WordNet interface JWNL [1] in order to use a database loaded with the latest information available on WordNet;

5.2 Future Work

In future work it would be interesting to:

- Adapt the bootstrap based system to work with NLP chain dependency patterns;
- Study the priority of the features of the answer that should be present on the distractors, increasing the number of distractors generated while keeping the quality of the generation;
- Study the adaptations required on the generation of question items for differently themed documents (for example, changes on the rules when the corpus is about historical facts);
- Create an user-friendly interface for the rules based system;
- Explore additional word classification/relation resources to improve results on both systems;
- Extend the interface to allow the use of the patterns discovered by the bootstrap component on the generation component;
- Extend the interface allowing the use of other corpus sources (for example text files given by the user);
- Create an evaluation and control module for the bootstrap-based solution that would allow an assessment of the contribution on each pattern to the system results;
- Application of an anaphoric reference solver component to improve the quality of the test items generated;
- Explore additional distractor sources (for example WordNet);
- Explore other types of multiple-choice test items (for example word-bank questions).

[1] http://sourceforge.net/projects/jwordnet/

Bibliography

Baptista, J., Costa, N., Guerra, J., Zampieri, M., Lurdes Cabral, M. de, & Mamede, N. (2010). P-AWL: Academic Word List for Portuguese. In *Computational Processing of the Portuguese Language* (p. 120-123). Porto Alegre, RS, Brazil: Springer.

Brown, J. C., Frishkoff, G. A., & Eskenazi, M. (2005). Automatic question generation for vocabulary assessment. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing* (pp. 819–826). Morristown, NJ, USA: Association for Computational Linguistics.

Chen, C.-Y., Liou, H.-C., & Chang, J. S. (2006). FAST: an automatic generation system for grammar tests. In *Proceedings of the COLING/ACL on Interactive presentation sessions* (pp. 1–4). Morristown, NJ, USA: Association for Computational Linguistics.

Correia, R. (2010). *Automatic question generation for reap.pt tutoring system*. Unpublished master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa, Portugal.

Fellbaum, C. (1998). *Wordnet : an electronic lexical database*. Cambridge, Mass : MIT Press.

Graesser, A., Rus, V., & Cai, Z. (2008). Question classification schemes. In *Proceedings of Workshop on the Question Generation Shared Task and Evaluation Challenge.* Arlington, VA, USA.

Hagège, C., Baptista, J., & Mamede, N. (2008). Reconhecimento de entidades mencionadas com o XIP: Uma colaboração entre a Xerox e o L2F do INESC-ID Lisboa. In *Desafios na avaliação conjunta do reconhecimento de entidades mencionadas: O Segundo HAREM.* (p. 261-274).

Hoshino, A., & Nakagawa, H. (2005). Webexperimenter for multiple-choice question generation. In *Proceedings of HLT/EMNLP on Interactive Demonstrations* (pp. 18–19). Morristown, NJ, USA: Association for Computational Linguistics.

Hoshino, A., & Nakagawa, H. (2007). Assisting cloze test making with a web application. In *Proceedings of Society for Information Technology and Teacher Education International Conference* (pp. 2807–2814). San Antonio, Texas, USA.

Ignatova, K., Bernhard, D., & Gurevych, I. (2008). Generating high quality questions from low quality questions. In *Proceedings of Workshop on the Question Generation Shared Task and Evaluation Challenge.* Arlington, VA, USA.

Judge, J., Cahill, A., & Genabith, J. van. (2006). 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* (pp. 497–504). Morristown, NJ, USA: Association for Computational Linguistics.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, *10*(8), 707-710.

Li, X., & Roth, D. (2002). Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics* (pp. 1–7). Morristown, NJ, USA: Association for Computational Linguistics.

Lin, C.-Y. (2008). Automatic question generation from queries. In *Proceedings of Workshop on the Question Generation Shared Task and Evaluation Challenge.* Arlington, VA, USA.

Liu, C.-L., Chun-HungWang, Gao, Z.-M., & Huang, S.-M. (2005). Applications of lexical information for algorithmically composing multiple-choice cloze items. In *Proceedings of the 2nd Workshop on Building Educational Applications Using NLP* (pp. 1–8). Association for Computational Linguistics.

Marujo, L., Lopes, J., Mamede, N., Trancoso, I., Pino, J., Eskenazi, M., et al. (2009). Porting REAP to European Portuguese. In *Proceedings of the SLaTE Workshop on Speech and Language Technology in Education.* Brighton, UK.

Mitkov, R., & Ha, L. A. (2003). Computer-aided generation of multiple-choice tests. In *Proceedings of the First Workshop on Building Educational Applications using Natural Language Processing* (p. 17-22). Morristown, NJ, USA: Association for Computational Linguistics.

Mitkov, R., Ha, L. A., & Karamanis, N. (2006). A computer-aided enviroment for generating multiple-choice tests items. *Natural Language Engineering*, *12*(2), 177-194.

Petrov, S., & Klein, D. (2007). 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* (pp. 404–411). Rochester, New York: Association for Computational Linguistics.

Ruppenhofer, J., Ellsworth, M., Petruck, M. R. L., Johnson, C. R., & Scheffczyk, J. (2006). *Framenet II: Extended theory and practice*.

Santos, D. (2010). *Extracção de relações entre entidades*. Unpublished master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa, Portugal.

Silva, J. (2009). *Qa+ml@wikipedia&google*. Unpublished master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa, Portugal.

Silveira, N. (2008). Towards a framework for question generation. In *Proceedings of Workshop on the Question Generation Shared Task and Evaluation Challenge.* Arlington, VA, USA.

I Appendices

A Rule's file used on the second evaluation

The following rules were used for the second evaluation of the rules based system presented on this thesis.

```
Building-Location(*[monument];*[location])
        Q: Onde pode ser encontrado #1?
        A: #2
|
Building-Location(*[company];*[location])
        Q: #1 tem a sua sede em?
        A: #2
|
Business[employee](*[people,individual];*[company])
        Q: Em que empresa trabalha #1?
        A: #2
|
Business[employee](*[human];*[company])
        Q: Em que empresa trabalha #1?
        A: #2
|
Business[profession](*[people,individual];*[profession])
        Q: #1 é conhecido por ser?
        A: #2
|
Business[profession](*[human];*[profession])
        Q: #1 é conhecido por ser?
        A: #2
|
Business[founder](*[people,individual];*[company])
        Q: #1 fundou?
        A: #2
```

```
|
Business[founder](*[human];*[company])
        Q: #1 fundou?
        A: #2
|
Business[founder](*[people,individual];*[org])
        Q: #1 fundou?
        A: #2
|
Business[founder](*[human];*[org])
        Q: #1 fundou?
        A: #2
|
Lifetime[born](*[people,individual];*[date])
        Q: #1 nasceu em?
        A: #2
|
Lifetime[born](*[human];*[date])
        Q: #1 nasceu em?
        A: #2
|
Lifetime[death](*[people,individual];*[date])
        Q: #1 morreu em?
        A: #2
|
Lifetime[death](*[human];*[date])
        Q: #1 morreu em?
        A: #2
|
People-Location[residence](*[people,individual];*[location])
        Q: #1 reside em?
        A: #2
|
People-Location[residence](*[human];*[location])
        Q: #1 reside em?
        A: #2
```

```
|
People-Location[place-of-birth](*[people,individual];*[location])
        Q: #1 reside em?
        A: #2
|
People-Location[place-of-birth](*[human];*[location])
        Q: #1 reside em?
        A: #2
|
People-Location[country-of-birth](*[people,individual];*[gentcountry])
        Q: Qual a nacionalidade de #1?
        A: #2
|
People-Location[country-of-birth](*[human];*[gentcountry])
        Q: Qual a nacionalidade de #1?
        A: #2
|
People-Location[country-of-birth](*[people,individual];*[country])
        Q: #1 nasceu em?
        A: #2
|
People-Location[country-of-birth](*[human];*[country])
        Q: #1 nasceu em?
        A: #2
|
Family[spouse](*[people,individual];*[people,individual])
        Q: #1 casou-se com?
        A: #2
|
Family[spouse](*[human];*[people,individual])
        Q: #1 casou-se com?
        A: #2
|
Family[spouse](*[people,individual];*[human])
        Q: #1 casou-se com?
        A: #2
```

```
|
Family[spouse](*[human];*[human])
        Q: #1 casou-se com?
        A: #2
|
```