A Web-Based System for Automatic Language Skill Assessment: EVALING

Cédrick Fairon
Laboratoire d'automatique documentaire et linguistique (LADL)
University of Paris 7
2, Place Jussieu (CASE 7031)
75251 Paris CEDEX 05
France
fairon@ladl.jussieu.fr

Abstract

EVALING is a Leonardo da Vinci project funded by the European Union, involving four European laboratories[1]. The aim of the project is to build an automatic system to evaluate language skills in people's native language. This paper focuses on native French. Other partners are working on their own language and are building specific tests (Italian and German). EVALING is an 'Item Banking'[2] system: exercise database allowing dynamic design of questionnaires. We present a technique based on the use of NPL tools that assure easy and costless updating of these databases. In addition, we underline the interest of Local Grammars (Finite State Transducers) for scoring exercises on language.

Introduction

EVALING is a Leonardo da Vinci project funded by the European Union, involving four European laboratories. The aim of the project is to build an automatic system to evaluate language skills in people's mother language. Each partner is working on his own language and building specific tests (at the present: French, Italian and German). In this paper we will present French.

The first step consists, for each language, in determining the fields to be tested and the types of exercises which can be computerized to carry out this task. We have observed this task differs from one language to another. For example, spelling exercises are relevant in French, but they are not very interesting for German, since people make few mistakes. For French, we decided to focus on syntax, lexicon, spelling and reading comprehension. Hence, we oriented the reflection on the tests that could be automatized and on those that could not. At this point, automatization in the EVALING sytsem bears on three phases of the evaluation process:

- dynamic setting up of tests (exercises are stored in tables of an exercise database). We assume that if a person has to take the test more than once, this person should not get the same set of questions twice,
- automatic grading of tests (including storage of marks in a client database),
- semi-automatic filling of exercise databases (with the assistance of linguistic tools).

First, we will discuss some technical aspects: EVALING is a Web-based program interacting with large exercise databases and a client database. We will explain how this ' item Banking ' system has been implemented on a Web Server as an ISAPI (Internet Server Application Programming for Microsoft Information Server[3]) and how exercise databases were built. This last point is a key issue, because the aim is not so much to gather a fixed amount

[1] Association pour le traitement automatique des langues (ASSTRIL) for French, Consorzio Lexicon Ricerche from the University of Salerno, for Italian & Pädagogische Hochschule Karlsruhe and Universität München for German.

[2] "Item banking covers any procedures that are used to create, pilot, analyse, store, manage and select test items so that multiple test forms can be created from a subset of the total 'bank' of items. " Brown (1997).

[3] Chapman (1997) discuss advantages and disadvantages of ISAPI programming..

of exercises, but rather to be able to renew them easily. We designed a set of linguistic tools to satisfy this demand. The set of tools is based on the software INTEX, developed at the LADL by Max Silberztein[4]. Of course, it is not always possible to automatize the creation of exercises. In certain cases, the work will have to be done manually.

Second, we will present the 'Administrator side'. We call 'administrator' the person or team who needs to evaluate a large group (students, employees, applicants, etc.). An interface enables the administrator to define the parameters of the test (length, level, etc.) and to perform some statistical analysis on the client database.

Finally, we will deal with the 'client side'. The client registers himself and then has access to the test session through a Web-Client software. All test forms are in HTML form.

1. The Developer side

1.1 The Main Program and the Databases

From a technical point of view, EVALING uses a Multithreaded Automation Object. This technology allows multiple threads of execution (Apartment-model threading). Thus, the EVALING kernel is an 'Active X DLL'. In the DLL, all exercises are designed on a single module which makes it easier to develop and manage the whole system. In fact, in order to add or remove exercises, we just have to add or remove modules and to change a global variable in the main module which indicates to the system the number of exercises and their order in the session. The system is developed to run with *Microsoft Information Server* and works as a server application.

Sets of exercises passed down to a client are dynamically composed by the system at the time of the user's request. All exercises are stored in tables of the Exercise database. For each questionnaire, the system extracts a set of exercises from a table. Each record in the table corresponds to one sentence, or one short text. Therefore, several records are retrieved by the system to constitute the questionnaire that will be sent to the client.

Corrections of the user's answers are provided by JavaScript programs that use data of hidden fields in HTML form or by more complex programs running on the server when necessary. When we make use of JavaScript programs to correct a form, we never display answers clearly in the JavaScript source (functions that take ASCII values or binary values evaluate the validity of answers). Answers are therefore not readable in HTML source code. This practice is a second level of protection, because the whole test session occurs in a Browser Window that does not contain the menu bar that allows the view source action.

It is imperative that sets of exercises have the same level of difficulty from one client to another. To signal this stability, a field in each record indicates the level of difficulty. When the EVALING system composes a questionnaire, records are chosen in such a way that the sum of the difficulty marks is always the same (this sum is given by the administrator who can choose the global level of difficulty). Grading occurs at the development time. A script, which uses linguistic tools (INTEX programs, local grammars, dictionaries and linguistic descriptions) evaluates the sentences. For French, the difficulty of a sentence depends on:

-length in words,
-presence/absence of negation, conjunction, relative pronoun,
-lexical difficulty: to measure this parameter, we use an electronic dictionary: the DELAF dictionary for French that contains more than 900.000 inflected forms. It is divided into three levels of difficulty: common and easy words constitute the first level, technical and unusual words the third level and in between, we have a second level consisting of words whose understanding is hazy.

The dynamic design of a questionnaire requires large databases, with a large number of sentences for each exercise. It is costly to produce all of them manually. To avoid such a difficulty, we designed a set of search tools that apply to large corpora and retrieve sentences or

[4] INTEX is a parsing system based on wide coverage dictionaries and local grammars (represented by Finite State Automata) which applies to lage corpora, that is 100Mo. Cf. Silberztein (1993).

short texts that match a given linguistic pattern. These tools constitute a full-fledged parser based on the use of Finite State Transducers (FST)[5]. They use morpho-syntactic information provided by wide coverage electronic dictionaries[6], allowing us to retrieve accurate linguistic information[7].

The software INTEX includes a graphic editor (FSGraph) which allows ergonomic designing of FSTs (which are graphically represented as graphs). Linguists or other non computer scientists can easily design and maintain FST libraries. Graphs can be used in three modes[8]: 'simple mode' for retrieval purpose, 'fusion mode' for inserting patterns in the recognized sequence and 'replace mode' for substituting a pattern to the one matched by the graph. For example, the following graph, when used in 'simple mode', locates in a text the patterns displayed in the boxes (*inputs*) and, when used in 'replace mode', it substitutes to the input the text displayed below the boxes (*outputs*).

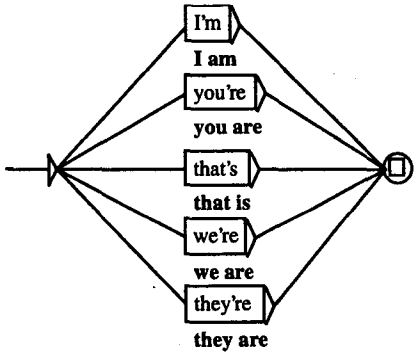

Fig. 1: Sample of graph

1.2 Types of exercises

We built two types of exercises:
First, we have a set of exercises whose correction consists of comparing the user answers to the solutions registered in the database. This elementary way of scoring is used for exercises where the number of different correct answers is limited (generally to one or two possibilities). User answers are evaluated as either right or wrong, with no intermediate scores. On the screen, the exercise can take the appearance of a multiple choice test (if we add distractors) or of a form to complete (if we remove from the original sentence patterns to be tested).

Our methodology produces this kind of exercise easily and economically. Since we extract the items for these exercises from corpora, all items we put in the database are well-formed items[9]. Thus, the validity check of the users answers can be a simple string comparison. The example below underline this simplicity (cf. *1.3 Sample of exercise development*).

Second, we are working on the conception of exercises where expected answers are more free, but not completely. In this case, scoring is performed by means of specialized grammars. Technically, these grammars are also Finite State Transducers. They describe all possible structures of a given context. Because these grammars are specialized to the descriptions of limited contexts, they are called 'local grammars'.

Local grammars introduce a lot of flexibility for rating exercises. We are testing two ways of scoring: binary scoring (right/wrong) and multi-value scales. In the first case, we use a grammar in 'simple mode' to verify an answer: answer matched by the grammar is valid, otherwise it is considered wrong. In the second case, the grammar is used in 'replace mode': if a path of the grammar matches the answer, the output combined to the path is produced. This output can be a score or a formal information processed later by a scoring program. It is thus possible to produce different outputs (scores), and even outputs for each path of the graph. For example, this system allows us to rate an answer according to the linguistic register used by the

[5] NLP functionalities of these automata are discussed in Roche, 1997.

[6] For a description of dictionaries format and samples of application, see Courtois (1990).

[7] An example of graph library is avaible on the LADL Web server: recognition of English compound verbs: auxiliaries, modals, aspectuals etc. Cf.: http://www.ladl.jussieu.fr/tools/tools.html#lemma

[8] For a simple description, see Silberztein (1998).

[9] If we assume that the corpora is error free, but it is not. A human reader has to verify the items of the database.

testee or according to the amount of details given.

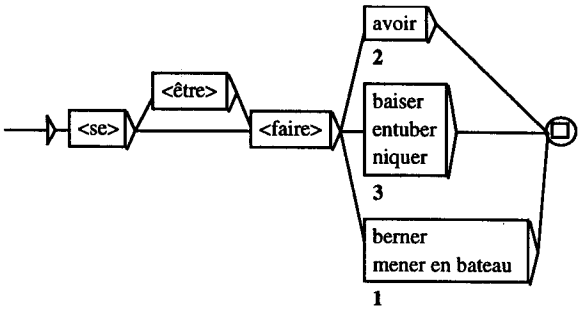

Fig. 2: Linguistic register

In figure 2, words between brackets are lemmas: <être> refers to all the forms of the verb *être* and <se> to the contracted form *s'* and *se* (this information is provided by dictionaries). In this way, the followings utterances are matched by the graph: *s'est fait avoir, se faisait avoir*, etc. The output number refers to the linguistic register: 1 indicates the best forms, 2 a spoken form and 3 slang terms.

1.3 Sample of exercise development

We present a simple case of exercise building. The purpose of this exercise is to test the ability to choose the right form of the French word *tout* ('all') that can be a noun, pronoun or adverb and that can be spelled *tout* , *tous*, *toute* or *toutes*.

1.3.1 Retrieving sentences with a graph

We design a graph and apply it to a corpus to retrieve sentences that match the sequences described by the graph. Then the graph is stored in the 'tool folder'. It will be reused later, when we will need to update the Exercises database.

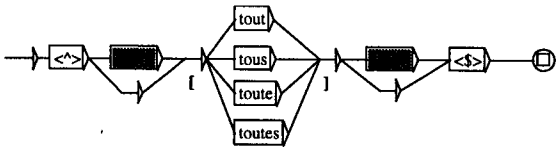

Fig. 3: Sample of locating graph

The graph is read from left to right (from the Initial node to the terminal one). If a sequence of words in the text is matched by one of the paths of the graph, the sequence is saved. An output is associated to this graph (represented below two boxes: "[" and "]"). We apply the graph in merge mode to insert this output in the text. Inserted signs are helpful to import sentences in the database.

Our corpus is preprocessed by INTEX. Since preprocessing segments the text into sentences, it is possible to refer to the 'beginning of a sentence' (<^> in our graph) or to the 'end of a sentence ' (<$> in our graph). The box containing 'MOT' is a link to another graph that represents a string of words (max. 10 words) and optional delimiters (apostrophe, comma, hyphen). This string is optional, that is why there is an alternative path under the box 'MOT'.

The following sentences are retrieved from a novel of Agatha Christie:

Son air très anglais avait **[tout]** pour séduire quelqu'un qui, comme moi, n'avait pas revu sa patrie depuis trois ans.

Je voudrais que, de retour chez vous, vous observiez le monde nouveau de l'après-guerre et que vous décidiez, en **[toute]** liberté et indépendance, ce que vous en attendez.

J'ai fait **[tout]** mon possible pour ne pas vous dire que je vous aime...

Et ils vécurent **[tous]** ensemble dans une petite maison biscornue.

1.3.2 Estimation of the sentence difficulty level

The tool is still under development and a study is going on to refine the criteria we are using. At this time, we consider three levels of difficulty: easy, medium and difficult. The level assessment depends on the length in number of words, presence of a modality (interrogative or not), lexical complexity and presence or absence of a negation, conjunction, relative pronoun. For example, the graph 'value.grf ' below detects negation, conjunction and relative pronouns. In this graph, GN is a link to another graph that describes summarily noun phrases. It guarantees that the pronoun which comes after GN is effectively a relative pronoun. NCR is a link to a graph that repeats the possibility to have a negation, conjunction or relative pronoun. In fact, a sentence can contain more than one pattern. <CONJC> and <CONJS> are tags that come from dictionaries and local grammars

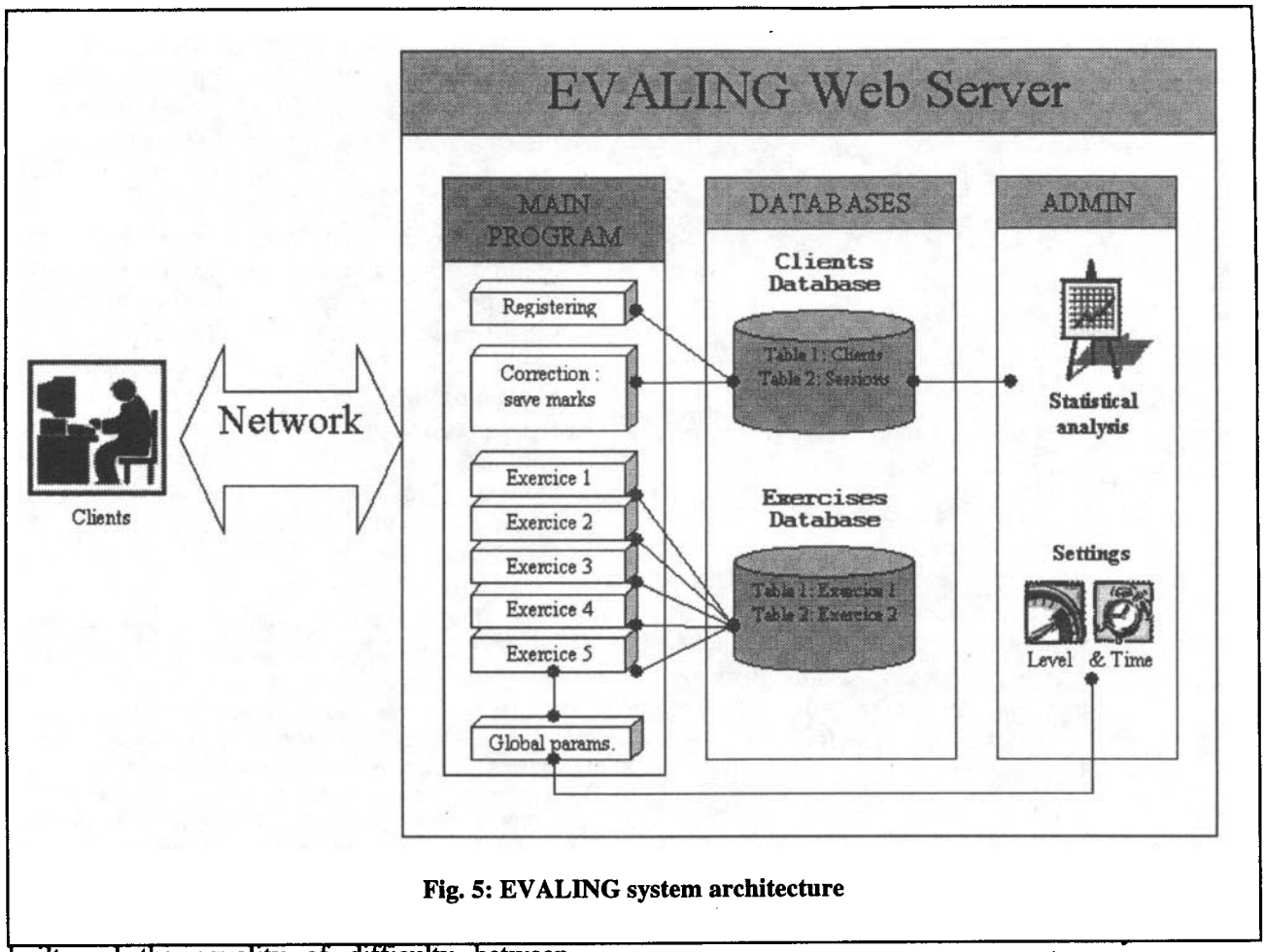

Fig. 5: EVALING system architecture

built and the equality of difficulty between questionnaires is tested by a function that refers to a key value. This key value is accessible to the Administrator who can decide to raise it or to reduce the global key value that gives the system the level of difficulty to compute. Since all marks are stored in a database, it is possible to perform some statistical analysis. Through the Administrator Interface, one can handle the usual numerical data: who got highest/lowest marks, where are the weaknesses/strengths of the group, etc. Statistical information is then displayed graphically and in tables. Example of application: In October 99, at the beginning of the academic year, EVALING will be used to help a language teacher with information about the specificity of this group of students. A statistical analysis of the marks obtained by about two hundred students is already a rich information. The administrator can detect common weaknesses and adapt his course to these difficulties.

returns the next exercise (e.g. exercises on past participles) to the client. At the present time, the test session is not 'adaptative'[10]: all clients answer the same questions (layouts used to compose sets of questions are the same for all users). At the end of the session, the client receives a report, in the case the Administrator has configured the system to do so. The report mentions only the marks registered by each exercise, without any other comments. Marks are also represented graphically by a Java Applet.

We were careful to avoid interferences between language skills and computer skills. Computerization should not interfere with language skill assessment. Our prerequirement was to create a system that could easily be used by someone not that familiar with computer handling. In fact, recent scientific studies

[10] Principles of Computer-Adaptative language Tests (CAT) are described in Tung (1986).

showed that there is little or no evidence of adverse effects on the computer-based tests due to lack of prior computer experience[11]. To ensure that, we decided to rule out technical manipulations like ' drag and drop ' and to make every effort to build a simple and intuitive interface. A tutorial is available from the start page on. Before getting into the test, the user can discover the different types of questionnaires and try each element of the answering system (in fact, any item in HTML Forms: pop-up menu, text field, button, radio button, etc). During the test a help button always displayed at the same location provides contextual information by means of a simple click.

4. Conclusion

The originality of our work lies in the use of powerful linguistic tools that can be adapted to a large variety of situations and that allow easy, fast and cheap renewal of the stock of exercises (with the aim of changing levels of difficulty or testing skills in technical languages).

Our experiments, are intended to show that local Grammars (FSTs) constitute a powerful tool for scoring exercises which have a wide range of valid responses.

References

BROWN, J.D. (1997) *Computers in language testing: present research and some and some future directions.* In "Language learning & technology", N°1, [http://polyglot.cal.msu.edu/llt/].

CHAPMAN D. (1997) *Wed Development with Visual Basic5.* Que Corporation, MacMillan, Indianapolis.

COURTOIS Bl. and SILBERZTEIN M., editors (1990) *Dictionnaires électroniques du français.* In "Langue française", N°87, Larousse, Paris, France.

ROCHE, E. and SCHABES Y., editors (1997) *Finite-State Language Processing,* MIT Press, Cambridge, Mass./ London

RUSSELL M., HANEY W. (1997) *Testing Writing on Computers: An Experiment Comparing Student Performance on Tests Conduced via Computer and via Paper-and-Pencil.* In "Education Policy Analysis Archives", Vol. 5. N°3, [http://olam.ed.asu.edu/epaa/arch.html].

SILBERZTEIN M. (1993) *Dictionnaires électroniques et analyse automatique de textes. Le système INTEX.* Masson, Paris, France.

SILBERZTEIN M. (1998). *Transducteurs pour le traitement automatique des textes.* In "Travaux de linguistique". N°37. Duculot, Bruxelles, Belgium, pp. 127-138.

TAYLOR C., JAMIESON J., EIGNOR D. and KIRSCH I. (1998) *The Relationship Between Computer Familiarity and Performance on Computer-based TOEFL Test Tasks.* Education Testing Service, Princeton.

TUNG, P. (1986) *Computerized adaptive testing: Implications for language test developers.* In "Technology and language testing", TESOL, Washington.

[11] Taylor (1998) for the TOEFLE study and Russell (1997) for a comparison on multiple-choice tests.