UNIVERSIDAD POLITECNICA DE VALENCIA

DEPARTAMENTO DE SISTEMAS INFORMATICOS Y COMPUTACION

Master's Thesis in
Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Implementation of a Human – Computer Interface for Computer Assisted Translation and Handwritten Text Recognition

by Jorge Carlos Ocampo Sepúlveda
supervisor: Enrique Vidal Ruiz
Valencia, February 25, 2009

Abstract

A human-computer interface is developed to provide services of computer assisted machine translation (CAT) and computer assisted transcription of handwritten text images (CATTI). The back-end machine translation (MT) and handwritten text recognition (HTR) systems are provided by the *Pattern Recognition and Human Language Technology (PRHLT)* research group. The idea is to provide users with easy to use tools to make interactive translation and transcription feasible tasks. The assisted service is provided by remote servers with CAT or CATTI capabilities. The human-computer interface keeps track of the CAT / CATTI suggestions and allows the user to modify these suggestions through a text editor.

The human-computer interface communicates the user with the CAT / CATTI server. The CAT remote server engines accept translation requests from a source language. The user can validate a prefix from the translated sentence, and the server provides the most probable completions compatible with the source sentence. The CATTI remote server engines accept requests of text transcription from images. The user can validate further prefixes and pose requests for completion. In addition to this, the interface supplies the user with tools for efficient local edition. The human-computer interface allows the user to visualize the required line images of the document to the person is working with, along with the corresponding CATTI predictions. The user can validate a prefix from a prediction, and the server provides the most probable completions compatible with the source image. The user can validate new prefixes and pose any requests for completion. The tools for efficient local edition are available for transcribed text too.

Table of Contents

List of Figures

1 introduction

This report is the result of a work carried out at the *Pattern Recognition and Human Language Technology (PRHLT)* research group. *PRHLT* researchers belong to the Departamento de Sistemas y Computación (DSIC) and the Instituto Técnico de Informática (ITI) of the Universidad Politécnica de Valencia. The *PRHLT* research group pursues several investigative fields, some of them in the field of machine translation (MT) technology and handwritten text recognition (HTR).

The fields of MT and HTR in the world have reached some degree of success, but are yet far from perfection. For that reason it is necessary human supervision and correction of the results of the translation and transcription processes. The process of supervision an correction can be speed up by providing an agile interaction between the user and the translation device. The *PRHLT* research group has made some investigation in that direction and experimental MT software able to give output suggestions has been developed. The same has been done in the field of HTR.

The proposed task was to develop a graphical human-computer interface (HCI) prototype for the software developed by the *PRHLT* research group. There are previous works, indeed. An assisted translation interface was developed in Java[1] and an assisted HTR demo interface exists as a web site at http://katti.iti.upv.es/ [Casacuberta el al.]. In our case, the purpose was to create an integrated interface both for assisted translation and assisted HTR apt to be used in multiple platforms like Linux and Windows using the GTK tool kit.

1 Interfaz para el sistema de ayuda a la traducción del PRHLT. María Teresa González Cavero.

2 Basic Concepts

2.1 Machine Translation (MT)

In this global world, interaction between different languages and people of different cultures is an everyday event. People of different countries try to communicate their ideas. Not all people like learning another language. They would prefer to be told what they want to hear in their own language.

Translation is a business demanded everywhere, but expensive and scarce. Human resources training is slow and demanding. Translation is far from the purse of common people. Solution to this was to create translation machines enabled to be used for every one and every where.

Machine translation results are yet liable to be improved. The solution of errors generated could be costly and time consuming. For that whatever method to reduce time and costs of qualified hand work is welcomed.

The advance of the translation science is constant but slow, and its application in its pure state is something that will take a time to be attainable. For that reason, computer assisted edition is an option achievable and economical.

There are many approaches to machine translation. Given a sentence x in a source language an equivalent sentence y in a target language is looked for. This is a problem without a unique solution. Under the statistical framework, models of the languages involved are used. The relation between the two

languages is modelled too. Among possible translations it is selected the one which maximizes

$$\hat{y} = \underset{y}{\operatorname{argmax}} \; P(y|x) \qquad (1)$$

Every sentence y in a target language is considered as a possible translation of any other sentence in another source language. Then, for every possible pair of sentences there is a probability $P(y|x)$. The target sentence is selected using the maximum posterior probability.

If we decompose $P(y|x)$ using Baye's rule we get:

$$\begin{aligned} \hat{y} = \underset{y}{\operatorname{argmax}} \; P(y|x) &= \underset{y}{\operatorname{argmax}} \; \frac{P(x|y)P(y)}{P(x)} \\ &= \underset{y}{\operatorname{argmax}} \, P(x|y)P(y) \end{aligned} \qquad (2)$$

This inverse approach is interpreted as the use of a target language model $P(y)$, and a alignment and lexical model $P(x|y)$ to discover which was the most likely original sentence *y* given its distorted version *x* observed through a “distorted channel”.

The output language model $P(y)$ is used to raise the probability that only valid sentences will result from the translation process. The language models commonly used are Word n-grams, N-grams of categories, Regular Grammars or Context-Free Grammars.

In the single-word based alignment models the alignment and lexical model $P(x|y)$ assume that an input word x_j, in position j, is generated by one target

word y_i, in position i. It is used an index variable to denote it, $a_j = i$. As a result, the translation probability can be broken down into a lexicon probability and an alignment probability. The words aligned are selected from a statistical lexicon, a kind of bilingual dictionary, $P(x_j|y_{a_j})$.

In the IBM Model 1, the alignment probability distribution used $P(a_j|a_1^{j-1}, J, y)$ is the Uniform Distribution, , that is all alignments have the same probability.

In the IBM Model 2, the alignment probability distribution used $P(a_j|a_1^{j-1}, J, y)$ is a zero-order alignment model where different alignment positions are independent from the others.

The Hidden Markov model, HMM, uses a first-order model $P(a_j|a_1^{j-1}, x_1^{j-1}, J, y) \approx P(a_j|a_{j-1}, J, y)$ where the alignment position a_j depends on the previous alignment position a_{j-1}.

the IBM Model 3, uses an inverted zero-order alignment model, $P(j|a_j, I, J)$, adding a fertility model $\Phi = \phi(y_i)$, that model the number of source words connected to a target word y_i.

In the phrase-based alignment models, the correspondences between word segments are modelled. A sequence of source words is aligned to a a sequence of target words, aiming at keeping the context. The basic unit is the phrase. The statistical dictionaries of single word pairs are substituted by statistical dictionaries of bilingual phrases.

Other possible decomposition of the direct probability is:

$$\hat{y} = \underset{y}{\operatorname{argmax}}\ P(y|x) = \underset{y}{\operatorname{argmax}}\ \frac{P(x,y)}{P(x)} = \underset{y}{\operatorname{argmax}}\ P(x,y) \tag{3}$$

That is, the joint probability is used. This approach is often implemented by means of stochastic finite state transducers. A SFT *T* is defined by (τ, P, P_F), where:

- τ = {Q, X, Y, q_0, Q_F, E} is a Finite State transducer with Q_F = Q.
- P y P_F are functions such that:

$$\sum_{(q',u,v,q)\in E} P(q',u,v,q) + P_F(q') = 1 \quad \forall q' \in Q \tag{4}$$

- Probability of a path, P_m, ending at the state q_m:

$$P(P_m) = \prod_{(q',u,v,q)\in P_m} P(q',u,v,q) P_F(q_m) \tag{5}$$

- Probability of a translation (x, y) of τ:

$$P_\tau(x,y) = \sum_{P_m \in P(\tau,x,y)} P(P_m) = \sum_{P_m \in P(\tau,x,y)} \prod_{(q',u,v,q)\in P_m} P(q',u,v,q) P_F(q_m) \tag{6}$$

P_τ(x, y) defines a joint probability distribution. The marginals defined by a Finite State Transducer τ are stochastic regular languages:

$$P_i(x) = \sum_{y\in Y^*} P_\tau(x,y) \qquad P_o = \sum_{x\in X^*} P_\tau(x,y) \tag{7}$$

These languages can be properly considered as *input* and *output* Languages Models corresponding to τ. These Languages Models are regular languages associated to the automata obtained by dropping the input and output symbols of each transition of the finite state transducer.

The machine translation model used by the server engine that will be

connected to the human-computer interface is the stochastic finite state transducer (SFST) model. But CAT edition can be implemented in any model, and using the proper socket interface, engines with any model, qualified to provide translation services or edition services could be connected to the interface implemented in this project.

2.2 Handwritten Text recognition (HTR)

Extract text from images is a needed work in the modern times. The demand for digitalizing documents is growing, not only because of the common development of everyday business, but also for the preservation of historical texts and for the spreading of human science and culture. Then, for these and others efforts, automated manners to perform text recognition tasks from images are welcome.

Very important and more difficult and demanding is the work of extracting handwritten text from images. While in printed texts characters are widely standard, in handwritten texts multiple possible sources make of it an overwhelming task. There are so many manuscript characters as human beings and fortuitous circumstances. Manuscript writing is a task commonly demanded from any human being in his relations with the government agencies, companies, banks, commerce, etc.

It would be crucial if the little efficient keyboard input system could be replaced by manuscript text recognizers.

The efficient use of the keyboard imply training, practice time and resources

that not every on can afford. But, despite the scientific efforts and investment, the development of a text recognition system free of errors and independent of the user is a distant perspective.

Handwritten Text Recognition (HTR) is a field of pattern recognition that deals with the transcription of cursive texts generally written by hand, into electronic (e.g. ASCII) form. HTR takes an image, in the form of an array of pixels spatially distributed, and try to interpreted it as a sequence of text characters that constitute a sentence in a determined language. Typically, the source image is submitted to preprocessing steps which provide a representation of it in terms of a sequence of feature vectors.

The general framework of Text Recognition can be expressed as follows: given an image presented as a set of vectors $\boldsymbol{x} = \langle x_{1,} x_{2,} \cdots \rangle$, there must be found a most probable sequence of words $\hat{w} = \langle w_{1,} w_{2,} \cdots \rangle$ by the application of a statistical rule, formulated by the next equation:

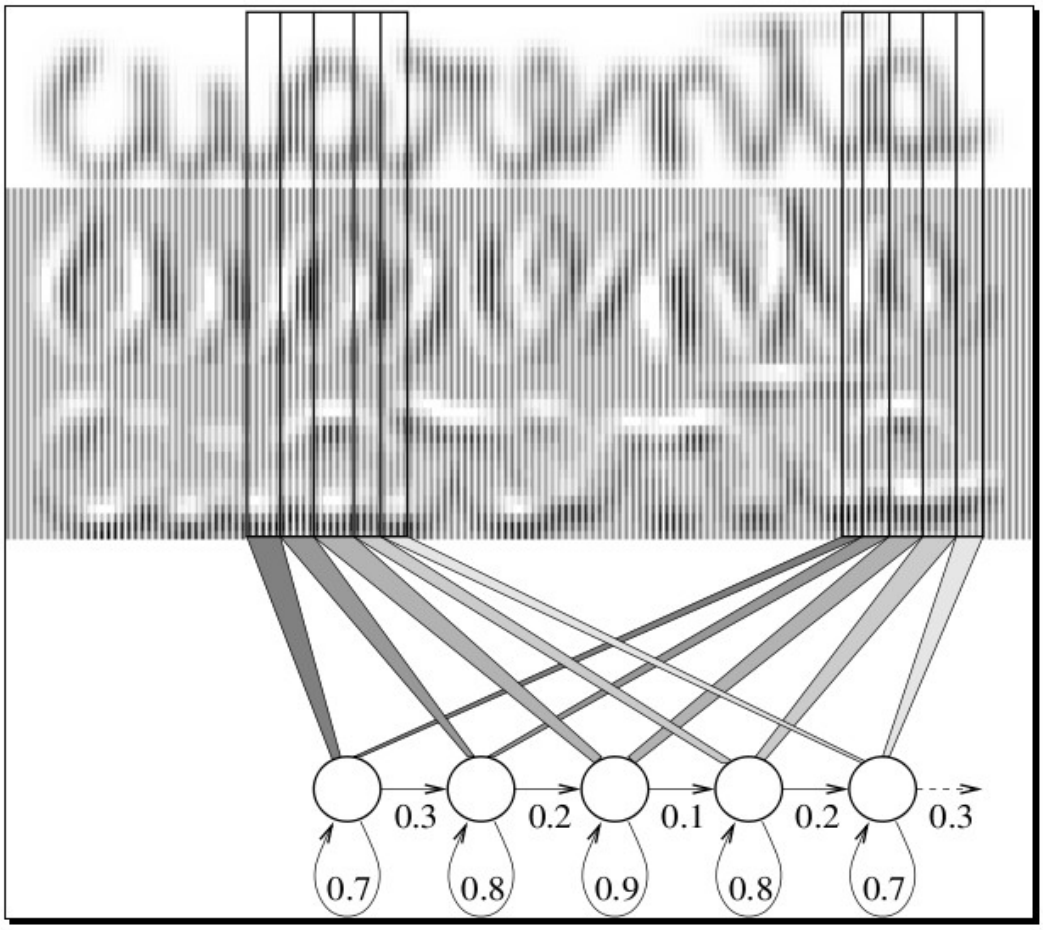
$$\hat{w} = \underset{w}{arg\,max} \; P(w|\boldsymbol{x}) \qquad (8)$$

Using the Bayes' Rule, the equation can be transformed:

$$\hat{w} = \underset{w}{arg\,max} \; P(\boldsymbol{x}|w)P(w) \qquad (9)$$

The first part, the inverted probability $P(\boldsymbol{x}|w)$, is the probability of observing the feature vector sequence $\boldsymbol{x}$, given a sequence of words w . Today in HTR this probability is modelled through morphological and lexical levels by Hidden Markov Models (HMM). In the morphological level characters are used as basic output units, generated by the input of feature vectors. In the

lexical level, words are made up of strings of characters. The HMMs must account for all the ways a word can be written, that is, upper case, lower case, marks, etc.



The second part of the equation, the prior $P(w)$, represents the syntactical level, modelled by a language model (LM). The result of the morphological level HMM must be screened by this LM, to increase the probability of output correctness, from a syntactical point of view. The LMs can be N-grams, SFST, etc.

Handwritten text is taken from an image. It must undergo some procedures in

the HTR process. This procedures can be summary as preprocessing, feature extraction and recognition.
Preprocessing objectives are various, such as to eliminate noise and background colour. Lines must be extracted in a process of detection and segmentation. Lines are normalize to correct line slope and text slant. Letters height and width must be determined.

Feature vectors are extracted from image, to obtain a sequence of real vectors.

2.3 Interactive – Predictive Processing

The multiple errors that plague machine nowadays translation and handwritten text recognition outputs can spoil an make useless the work obtained. But the advance of these technologies is not enough to ensure to be even near of a hundred percent correct result. Therefore manual proofreading and edition is always needed to produce documents of standard quality. However, manual corrections can be difficult and costly.

Interestingly, translation and transcription models capture a great deal of knowledge about the task they have been trained to. If you can ask adequately, this systems would answer things that were not firstly said but are correct and useful. If the human insight were used in addition to the computer translation and transcription models power, the results would be as correct as the ones from a human translator / transcriber, but less expensive.

Taking the correct fragments, validate them, and feedback the models with this knowledge, would increase the probability of obtaining a correct result. And if you have a mean to effectuate this process efficiently, with little amount of writing, in few time you can get a good result.

2.3.1 Interactive MT

Unfortunately, MT models are not perfect, and equation (1) has proved to yield results with grammatical errors, unnatural word order and other defects. Whether the models provide an acceptable result to the user it is a matter of the amount of post edition work needed to obtain what an expert translator could judge as acceptable.

Surely a person knowledgeable in translation could repair the result managing to obtain understandable sentences in the target language, bringing the output of the MT to the required degree of perfection. But, an expert would deem it preferable to translate directly without the help of machine translators If the post edition work is overwhelmingly high.

The Computer Assisted Translation, CAT, is a technique that tries to avoid post edition correction. It uses the language and translation models to complete a sentence known to be correct until a given point determined by the user. CAT language models uses input devices to interact to a human corrector, giving clues about correction of output sentences. The translation model used in the task of translation can be used now in the task of refinement of the phrases produced.

The polishing task can be performed in the moment the human assistant gives a sign about the point after which he judges the system results are not correct

or not desirable. The system takes the accepted section of the sentence and tries of obtain a set of correct completions of the sentence giving them as an option to the human.

Given the source sentence x, and a correct prefix p, a most probable suffix s that completes the sense of the sentence can be obtained as:

$$\hat{s} = \underset{s}{\operatorname{argmax}} \; P(s|x,p) \tag{10}$$

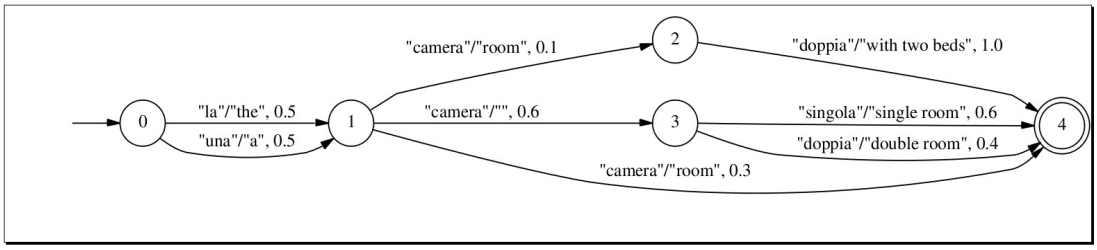
This equation can be rewritten as follows:

$$\hat{s} = \underset{s}{\operatorname{argmax}} \; P(ps|x) \tag{11}$$

The nature of the hint could vary. Be it the insertion of a character or a missing word, elimination of unnecessary words, or simply positioning the cursor. Mouse and keyboard are the devices commonly used for this purpose.

This CAT process is interactive. Once the human corrector has settled a prefix as acceptable, there begins a new assisted interchange in order to obtain a new prefix. In each stage the settled prefix becomes larger, with the use of few mouse and keyboard hits, until a suitable result is obtained.

Prefixes can be made up of complete words, in the case of CAT at the word level operation, or could end in a incomplete word, in the case of CAT at the character level operation.



[AERFAI]

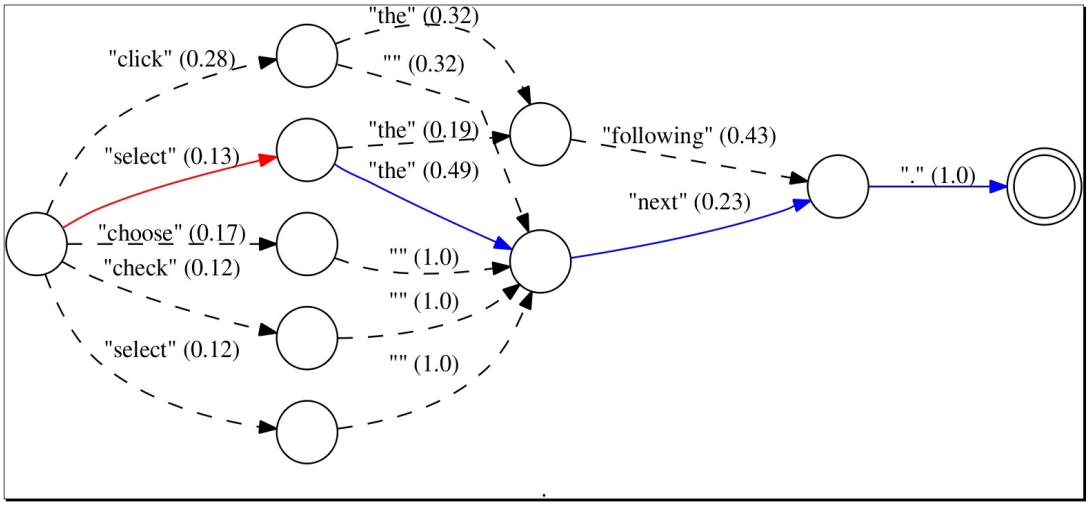
All the solutions to equation (1) from section **2.1**, for a given source sentence, *x*, can be thought of as a graph where each edge represents a target word. But the translation of a sentence is not unique. The edges of the graph considered have associated the probabilities assigned to the corresponding pairs of words (or phrases).

The suffix obtained as a result of the application of the equation (11) corresponds to the most probable path, in the graph corresponding to the language model, between the ones that pair the source sentence x and the several probable target sentences, made up of the prefix-suffix combinations.

When a prefix is settled, the search graph can be reduced to the edges and vertices corresponding to sentences which are consistent with the established prefix. Search can obtain a set of possible completions to be proposed to the human corrector for acceptance. Using this new reduced Word Graph, the *n*-best translations are searched for. The results are used for the user to obtain a new prefix and with user additional corrections, the new prefix can derive in a new Word Graph, leading to an interacting correcting process of increasing efficiency.

However, the selected prefix may be not present in the Word Graph. The solution to this difficulty is not taking the settled prefix p, but another prefix p'

that minimizes the edition distance to the settled prefix and being compatible with the Word Graph. Then, additional to the posteriori probability, a new factor is added to the translation, this is the edit cost between the target prefix and the approximated prefix found. It is equivalent to inserting fictitious edges to the search path, representing operations of deletion, insertion and substitution.



[AERFAI]

2.3.1 Interactive HTR

Computer Assisted Transcription of Text Images, CATTI, has been developed in a similar way to interactive Machine Translation. Pattern Recognitions systems yield results where errors are present. Post edition work is yet necessary, and it would be convenient to apply efficient methods intended to make it an economical and achievable task. For that reason equivalent techniques have been developed in the case of HTR. Interactive HTR uses input means to obtain feedback from a human corrector. The input means used range from mouse and keyboard to touch-screen devices and speech.

As compared with equation (8) in the CATTI statistical framework there is an

additional term, that represents the human feedback provided to the system. Then the HTR equation including this new element becomes:

$$\hat{w} = \underset{w}{arg\,max}\ P(w|\mathbf{x}, f) \qquad (12)$$

Where the *f* term represents the human feedback. The result in this case is an hypothesis generated by the system, the one that best fit the original facts and the user hints. The result $\hat{w}$ can be used as a new basis to provide further feedback to the system, which produces a new result. This interactive process ends when the user validates the result as correct.

The user feedback used in CATTI is the prefix validated by the user as correct. This prefix, in addition to the image used to produce the transcription, is used to originate one or more suffixes:

$$\hat{s} = \underset{s}{arg\,max}\ P(s|\mathbf{x}, p) \qquad (13)$$

The suffixes are searched in the space of all possible suffixes *s*, compatible with the validated prefix *p* and the image **x**. And the result must comply with the restrictions imposed by a language model and must be selected between the suffixes that can be written after the prefix *p*, as expressed by:

$$\hat{s} = \underset{s}{arg\,max}\ P(\mathbf{x}|p, s)P(s|p) \qquad (14)$$

The prefix and the suffix divide the image in two parts. The uncertainty about the bound *b* between both parts compels to consider multiple possible partitions, letting the process to select a best one, that is:

$$\hat{s} = \arg\max_{s} \max_{0 \leq b \leq m} P(\boldsymbol{x}_1^{\boldsymbol{b}}|p)\, P(\boldsymbol{x}_{\boldsymbol{b+1}}^{\boldsymbol{m}}|s)\, P(s|p) \qquad (15)$$

Where $\boldsymbol{x}_1^{\boldsymbol{b}}$ and $\boldsymbol{x}_{\boldsymbol{b+1}}^{\boldsymbol{m}}$ are the two fragments of a partition of $\boldsymbol{x}_1^{\boldsymbol{m}}$. The value *m* is the total number of image feature vectors. The search, then, can be restricted to the image areas corresponding to the possible suffixes and the prefix can be used to restrict the search space of the language model.

All the suffixes found in the “max” part of equation (15) can be seen as a word graph of words connected by edges, several edges leaving every node. A new word validated by the user signifies an advance in a path of the word graph and a smaller search space, resulting in increasingly correct prefixes, if not a correct sentence ready to be validated by the user.

As a matter of fact, the back-end CATTI engine to be used with the human-computer interface here developed, is implemented using word graph system for efficient search performance.

3 GTK Tool Kit

Some characteristics are desirable upon developing a software project. Graphic quality is important, but there are other features important not only to the user but to the developers. Portability, that is, the ability to operate in different platforms and systems, is an attribute that permits an application to reach a high number of users. It is very important that the tool kit used be actualized constantly and diligently. The components must be easy to use to the developers and to the users. In projects of this nature there is not other election but free software with an open source license. The developing tool must admit international set-ups. With these things in mind, and considering it is a widely and successfully used, the GTK tool kit was selected to develop the project.

GTK or the GIMP tool kit is a set of libraries to develop GUI (Graphic User Interface) applications for graphical environments. GTK is free software released under licence GNU LGPL, *Lesser General Public License*, cross platform, allowing development of free or proprietary software, without any licence fee. GTK was created to develop the GIMP image manipulation program. GTK+ was designed to let programming with languages like C, C++, C#, Java, Perl, PHP and Python.

GTK is based on several libraries of the GTK and GNOME team:

- Glib. Low level utility library, contains the basic structure of GTK and GNOME. It provides many data types, macros, string utilities, file utilities, etc.

- GTK. Libraries that define multiple objects and functions, useful to create an graphical user interface. It lets creation and use of windows, buttons, menus, labels, images, text editors, tool bars, etc.

- GDK. It is a graphical library used as intermediary between high level graphics and low level graphics. GDK stands for GIMPS Drawing Kit, originally developed on the X Window System, GDK lies between the X server and the GTK+ library, handling basic rendering such as drawing primitives, bitmaps, cursors, fonts, etc.

- Pango. Library for text design and rendering,
- Cairo. Library to let advance graphical rendering.

Some important pre-designed objects offered by the GTK library are:

- GtkWindow: This is the basic GTK widget. GtkWindow is top level widget which can contain other widgets. Main windows and dialogue windows can be implemented using this widget.

- Layout Containers: These are a set of widgets used in software graphical design. These containers can display any widget in particular dispositions and configurations. Widgets can be displayed in fixed sizes. But widgets can be set to fill the available space, or to expand and compete with other widgets for free space. These two possibilities are specially useful, because they permit the self organization of the widgets when the container window changes size, with no code writing. Some useful containers are listed here:

- GtkVBox: This container displays widgets in a column. Widgets are piled inside the vertical box on over the other in the adding order or are inserted in a given location.
- GtkHBox: This is a horizontal container. Widgets are organize in a single row. Widgets are packed from left to right in the adding order, or can be inserted in a given position.
- GtkTable: This container let the designer to organize the widgets in rows and columns. Widgets can be arranged in any pattern, aligned in relative position ones at the side of the others.
- GtkFixed: This container permits free layout of the widgets inside it. Widgets can be positioned in any fixed point and with fixed sizes. Widgets can be designed to overlap other widgets, a feature not permitted in other containers.
- GtkAlignment: This container manipulates the alignment of the widget inside any other container. It can manage the size of the child widget too.

In the graphical layout of a program, sometimes it is necessary to use various of these containers at the same time.

- GtkScrolledWindow: Scrolled windows are boxes that add scroll bars to child widgets. They act as containers, a widget is added to a scrolled window. Scroll bars operate automatically, they are shown or hidden if they are necessary. Scroll bars parameters self-adjust themselves to the circumstances.

- GtkEventBox: Some widgets don't have their own window, for example the image view. That disable them to catch any event. This box is

enabled to intercept any signal. It operates as a container, widgets are inserted inside the event box and fill it completely. Any event directed to the GtkEventBox can be managed an used to execute actions over the child widget.

- GtkIconView: This is a Widget which displays a list of icons in a grid. Icons can be associated to a label located down or at the right side of the icon. GtkIconView allows to select one or multiple items. Icons can be selected with the mouse pointer or using rubber-band selection, dragging the pointer. Keyboard shift and control keys can be used to select a range or a set of disconnected icons respectively.

- GtkTreeView: This object can display trees or lists. Grids and tables can be implemented using this Widget. GtkTreeView functions as a viewer of other objects. Columns, GtkTreeViewColumn type, independent objects, can be added to the GtkTreeView. Columns display cells and cells can render any useful information, such as a text string or an image.

- GtkTextView: This is a multi-line text editing widget. Text views are associated to an object that handle a text buffer, GtkTextBuffer. Text in GTK is in UTF-8 encoding, that is, one character can be encoded as multiple bytes. GtkTextBuffer must be used in the editing operations, in the positioning of the pointer (this because it changes a mark inside text buffer), and it has defined events related to this kind of operations (text changes, cursor insertion, etc.), while GtkTextView manage display operation and operations that don't involve change of text, like moving cursors (key pressing, mouse wheel rotation, cursor movement, etc.) .

- GtkImage: Image views are objects that display image buffers. These buffers can have origin in a file, this is managed by a GdkPixbuf; they can have origin in an animation using GdkPixbufAnimation; they can have origin in a canvas, that is, an off-screen drawable that supports standard drawing primitives, using GdkPixmap object; they can handle icons using GIcon or GtkIconSet; Pre-designed stock images, as standard icons (as “save as”, “copy”, etc.) can be handled too. GtkImage and it associated objects are highly automatized. With only one command can be presented images of types such as jpg, bmp, pgm, png, tif. Image buffers can be edited. But they do not have defined event signals. For this purpose, image views must be associated to event boxes, GtkEventBox objects. These let to use mouse events to handle clicking, dragging, etc.

- GtkLabel: This widget displays text not intended to be edited, but usually to label another widget.

- GtkMenuBar: The menu bar is used to display menu items, objects of GtkMenuItem type.

- GtkToolBar: The tool bar is used to display buttons of the GtkToolButton type. Every button can be associated to a click event, to fire the command it represents.

4 Software Layout

The software layout is developed around the basic capabilities offered for the CAT and CATTI edition technology to the users. Those basic capabilities are going to be offered through server connections to the *PRHLT* translation and handwritten text recognition engines. The user can open text and graphics files. Commands and interfaces are supplied in order to provide the basic file operations to the users. They can open files and save translation and text recognition results to new files. They can request connection to *PRHLT* servers. Servers can provide either basic translation and assisted edition operation as well as handwritten text recognition service and text recognition edition facilities.

The graphical interface is built in a GTK window, containing some easily distinguishable areas, some of them, like the menu bar, the tool bar and the status bar, standard in a window application. Container widgets were used to design the application layout. There are a wide variety of containers in GTK that let widgets to be organized in a automatic or semi-automatic way, specially in the operation of resizing the main window, without too much code writing.

The graphical interface was design to enable different components to interact ones with the others. Every component is made up of one or more widgets. Widgets events are used to let the user interact with the application. This is a list of some differentiated components in the application:

Explorer.
Connection Interface.
Translation Interface.

Text Recognition Interface.

Menu Bar.

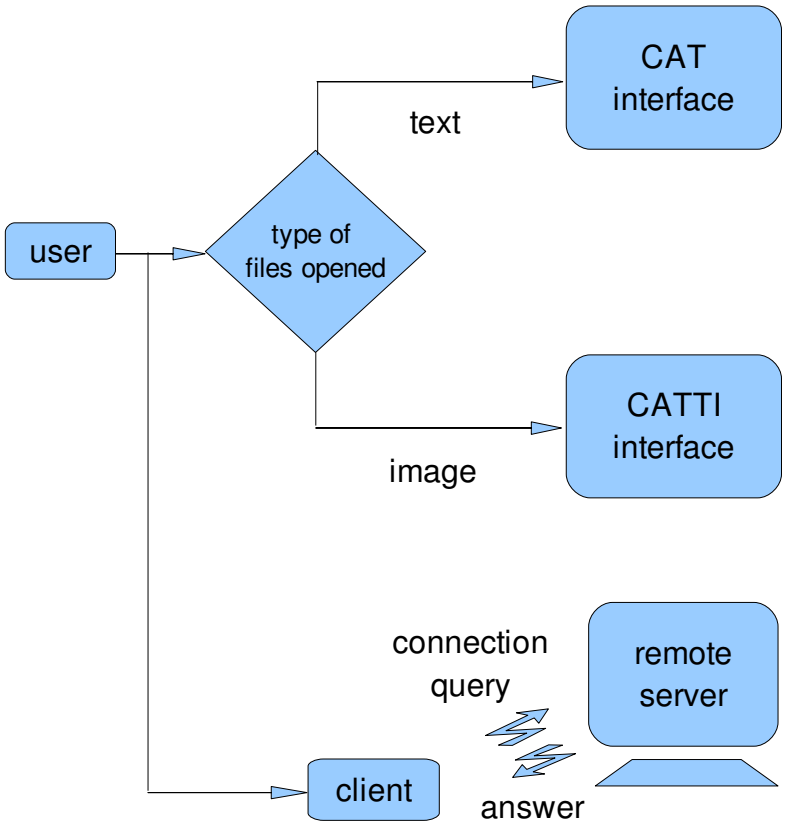
Tool Bar.

Status Bar.

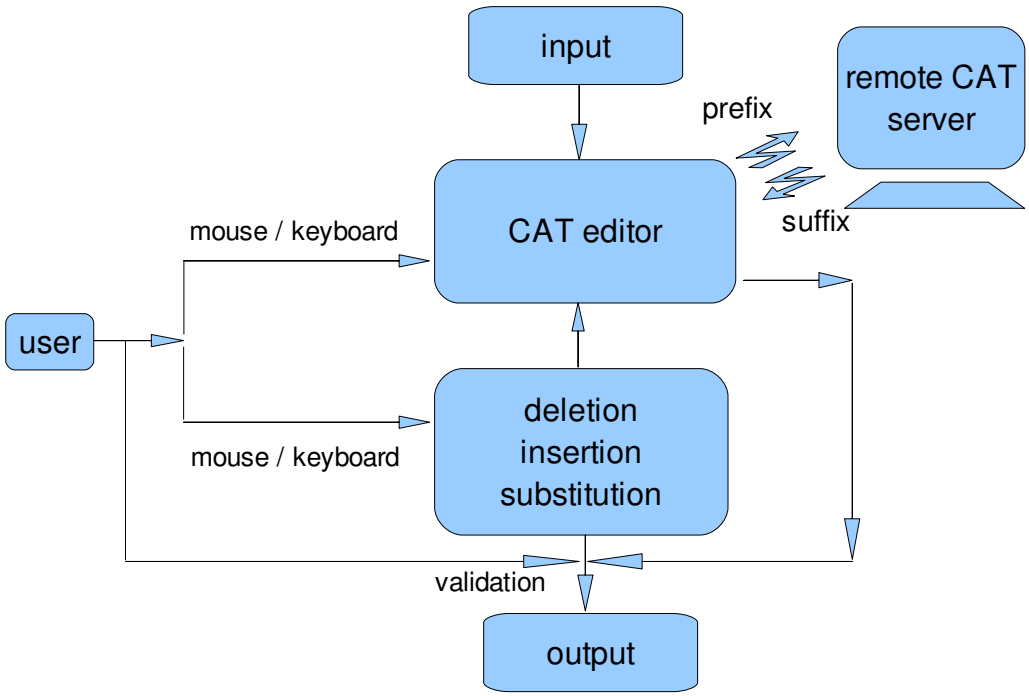
By default, the application shows an empty text translation interface. When files are opened by the user, the interface presented corresponds to the type of files opened.

4.1 Software Architecture

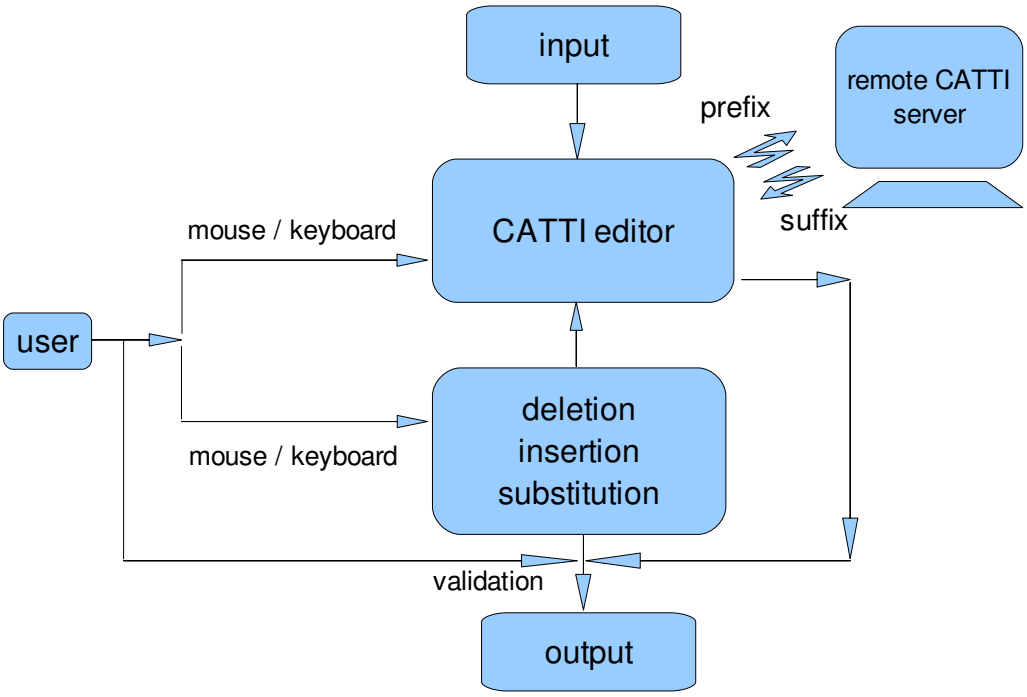
The software provides two major services, Computer Assisted Translation and Computer Assisted Transcription. Every one is performed on different kind of subjects, on text the former, on images the latter.



When the user selects a file, the file type is a clue to determine the type of operation he desires. The algorithm presents the CAT interface on opening a text file and the CATTI interface on opening a image file. The server connection is an independent operation. The user can connect to any server with independence of the type of file opened. Instead, requests are only permitted to the right server. A Computer Assisted Translation service can be only demanded to a CAT server, an a Computer Assisted Transcription can be only demanded to a CATTI server.



The input text files are transferred line by line into de editor. The user selects with the mouse or the keyboard the prefix to be sent to the remote server. The user can do local character edition operations too: deletion, insertion and substitution. The answer delivered by the remote server is submitted to the user, who can do further editions and remote requests interactively. The user validates the resulting sentence, and a new input is processed.

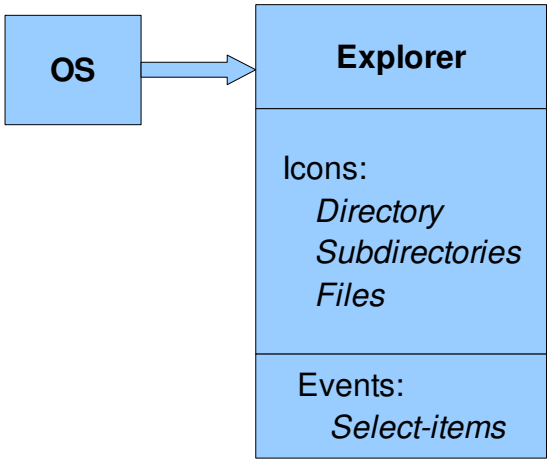


The handwritten text transcription editor has a similar design. The input subject of the HTR editor are image files. Each image is transferred to the server and it is transcribed. The resulting sentence can be fixed interactively with the help of the remote CATTI server. This server provides assisted transcription services. The user defines the settled prefix, a query is sent to the remote server, which provides possible completions. Results are presented to the user for validation. The user can execute correction on his own, using the edition capabilities; that is, character-level deletion, insertion and substitution. The resultant sentence of the process is validated by the user and a new input item is transferred to the editor.

4.2 Components

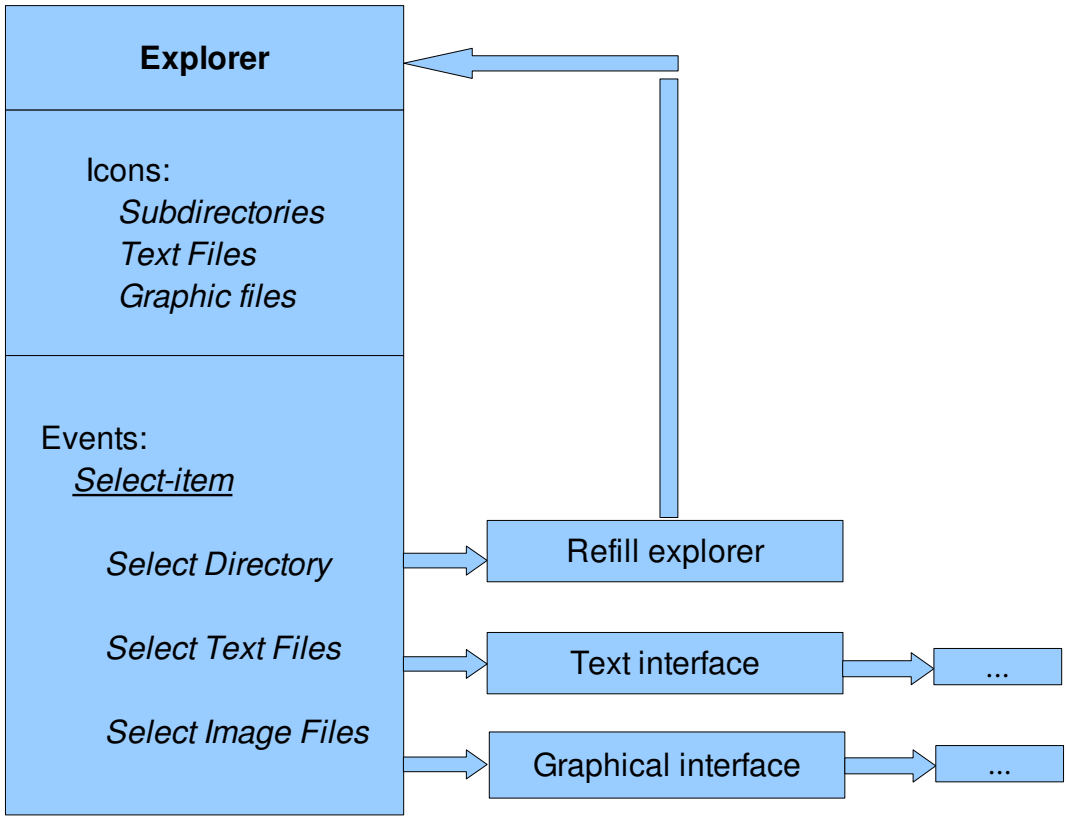
Now an overview of the components of the program is presented, along with a summary of their functions, interactions and events.

4.2.1 Explorer



The explorer takes a directory content and present it in the screen, in a explorer interface, made up of a GTK widget which displays a list of icons in a grid, identified as icon view. Items are presented in alphabetic order. Every element has its representative icon, taken either from the stock icons included in GTK pre-built common tool bar items library, or in the case of image files, an scaled image taken from the file itself.

Directories an files can be selected. Icon views can be set up to select one or multiple files . There are only one important *event* to be performed by the user on the explorer interface, called in GTK "item-activated"; this event is raised when the user selects one or more files and hit return or double click on the selected items:



- *On Select Items event*: If there are a mix of directories, text files or graphic files between the selected items, the function keeps the first file after sorting and determines the operation to be performed, according to the item type. If more than one directories are selected, the function takes the first one after sorting, and it is selected as *actual explorer directory*. If the fist item is a text file, text files are taken from the list and they are opened and are presented in the translation interface. If the fist item is a graphic file, every graphic file in the list is opened and they are presented in the text recognition interface.

  Select directory: when a directory is selected, its content is presented in the explorer interface. The explorer functions make use of the *dirent.h* library. It takes a directory as *actual directory* and consults the system

about the files and directories contained in the *actual directory*. Items are sorted in alphabetic order.

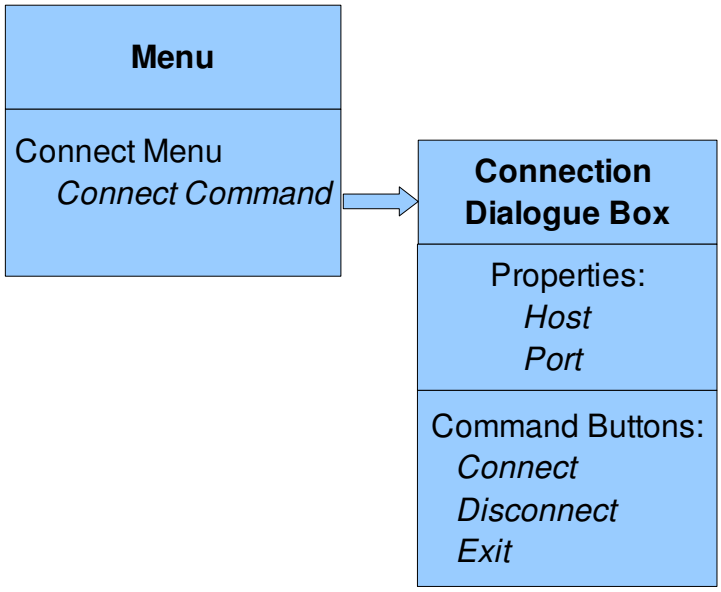
Select text file: when a file is selected, it is taken into the translation interface and presented in an input grid; every line of the is presented in a row of the input grid. If the line is made up of two sentences or strings, separated by a special character, such as '*#*', every string is presented in different columns. Text files are classified as such after a process have discarded them as directories (from the variable *d_type* of *struct dirent*, dp->d_type != DT_DIR), links (dp->d_type != DT_LNK), or graphics files (from extensions). Then text files and binary files remain. Binary files are filtered from the text files by a special function. The function opens every file classified in the group not belonging to image files. A repetition procedure is applied to every character of the file. Characters between *ascii* codes 0 and 31, named non-printable characters *NPC*, are counted, excepting HT (horizontal tabulation, *ascii* code 9 or '\t'), LF (line feed, *ascii* code 10 or '\r') and CR (carriage return, *ascii* code 13 or '\n') common in text files. If a character in the said range is found, the file is discarded as binary. If the file remains in the procedure up to the final, the file is accepted as text file and presented in the explorer with the icon assigned to this kind of files.

Select graphic file: Classification of selected image files. Every image is identified by his extension. Files that can be opened by the program are separated by the following extensions: *.jpg, *.bmp, *.pgm, *.png, *.tif. Every file name, after being classified, is maintained in a double linked list (a Glist). When every file selected has been processed, the lists of files are sorted in alphabetic order. Then, image files are opened,

scaled and presented as icons in the explorer interface, grouped by extension and ordered by name.

4.2.2 Connection Interface

Connection Interface use a socket to get a connection to a server providing services from translation to text recognition. For connections the program uses a *dialogue box* opened by a menu command. The server host and the port number must be introduced in the *dialogue box*. For the communication task, a C module provided by *PRHLT* group was used. The communication module implements basic socket connection operations, functions letting translation an



completion requests, and a function that permits handwritten text recognition operations and transcription edition.

Three command buttons can be selected:

- *The Connect Command* sends a connection request to a server, after the user has specified the host name and the port number. On

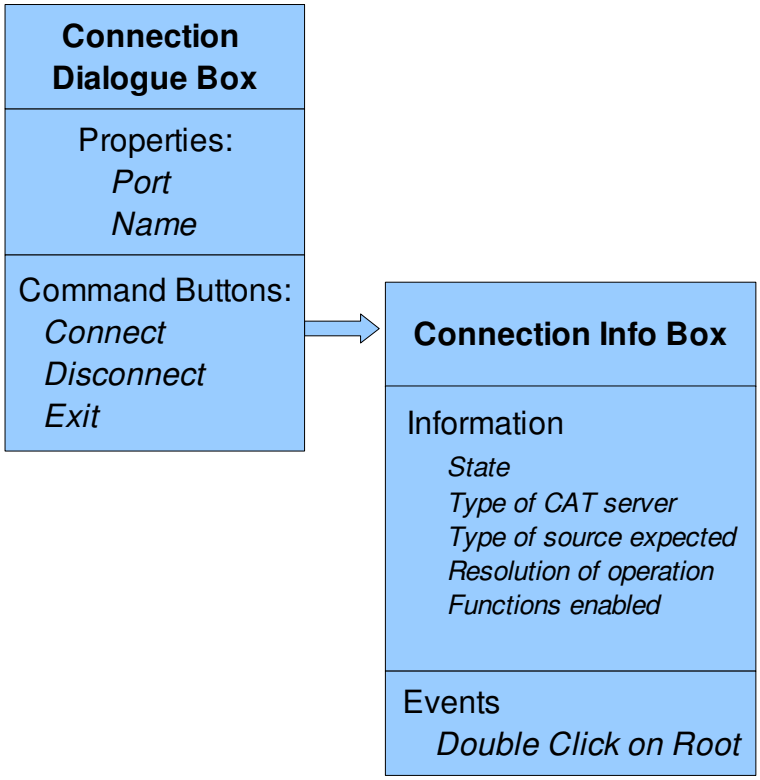
successful connection, another request bidding information on server capabilities is sent to the server. Server capabilities are shown in a GTK tree view box, the *connection information box*. After connection, the *dialogue box* is closed automatically.

- The Disc*onnect* C*ommand* closes a previous connection. The *connection information* box is refreshed to show the disconnected state .
- *The Exit* C*ommand* closes the *connection dialogue box* without changing the connection state.

After a successful connection request, a permanent box, a GTK tree view box, is opened inside the main window, with the connection information. The information provided is:

- Host name an port number.
- State of connection: connected, disconnected.
- Type of CAT server information indicates the basic task the server can develop; the possible options supported are the following: the server is a cat translation server, the server is a cat speech transcription server, the server is a cat handwritten transcription server. The speech transcription interface is not implemented in this software project at present.
- Type of source expected; indicates how the string should be used as source setting: the server takes the source as a source sentence, the server takes the source as a word graph file, the server takes the source as a feature vector file, the server takes the source as an image file, the server takes the source as an audio file. Provide interface for the last option, audio files, is not contemplated in this project at the moment.

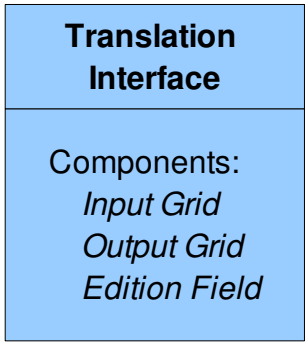
- Resolution of operation, that is, if the server allows char level operations or word level operations.
- Functions enabled: set source function, provide completions function, set prefix function, set prefix with on line handwritten text recognition function, set prefix with a generic feature vector, set prefix with generic features from a plain ascii text.



The Double Click event on the root of the C*onnection Info Box* opens the C*onnection Dialogue Box*.

4.2.3 Translation Interface

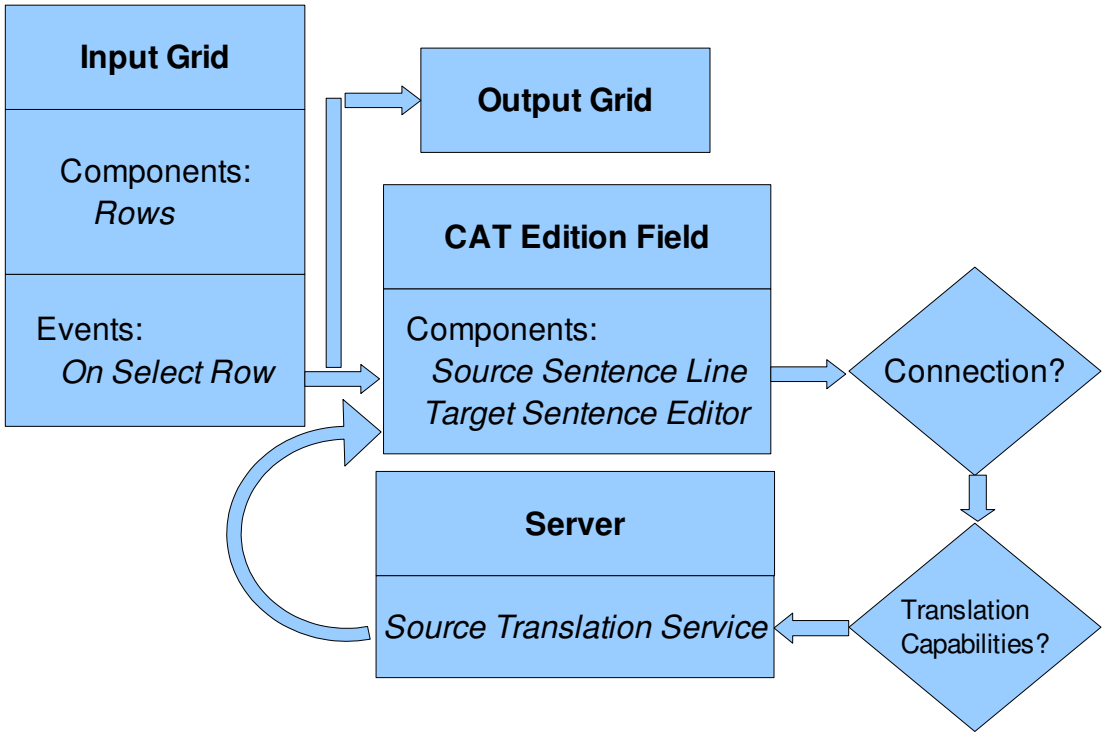
The Translation Interface is intended to implement machine translation services, applied on sentences, using the connected server capabilities. For this purpose we use an input grid, and an output grid, every one made up of two columns. Both columns can contain any number of cells enabled to render text. Text from the source file is allocated in lines, one in each cell, in the first column of the input grid. Every line is processed one at a time. After translated and corrected, the line being in the stage of process is transferred to the output grid. The text in the source language is placed in the first column of its line in the output grid and the translated text in the target language is placed in the second column. When a line is selected to be processed, it is sent to an edition section, a field composed of two lines. The upper line receives the source line. The lower line is a text editor aimed to contain the translated text and to accept the user-computer correcting interaction .



The type of files used in the Translation Interface are textual ones. Files presented in the navigator are text files or image files. To use the Translation Interface the user must open a text file. It is supposed that text files are arranged into lines, every one with a sentence in the language that the server connection accepts as source language. No error is raised if the language of

the source sentences is different from the one of the language model used by the remote server, but unexpected results will result. As an option, a second sentence can be in the line, divided from the former one with a conventional divider character such as the character "*#*". When the divider character is found in a line, the former part is allocated to the first column of the input grid, and the latter is allocated to the second column of the input grid.

4.2.3.1 Input Grid



Grids are assembled in GTK using Tree Views (GtkTreeView widget). As its name imply, a Tree View is a widget used to present items, that is, nodes, structured as a tree. But Tree Views were designed to perform multiple tasks. One of them is the construction of grids. For that purpose, Tree Views have a set of independent but related widgets and objects, like columns (GtkTreeViewColumn), cell renderers (GtkCellRenderer), etc. Columns are

created apart and then they are appended to the Tree View. Cells can be added at run time as needed.

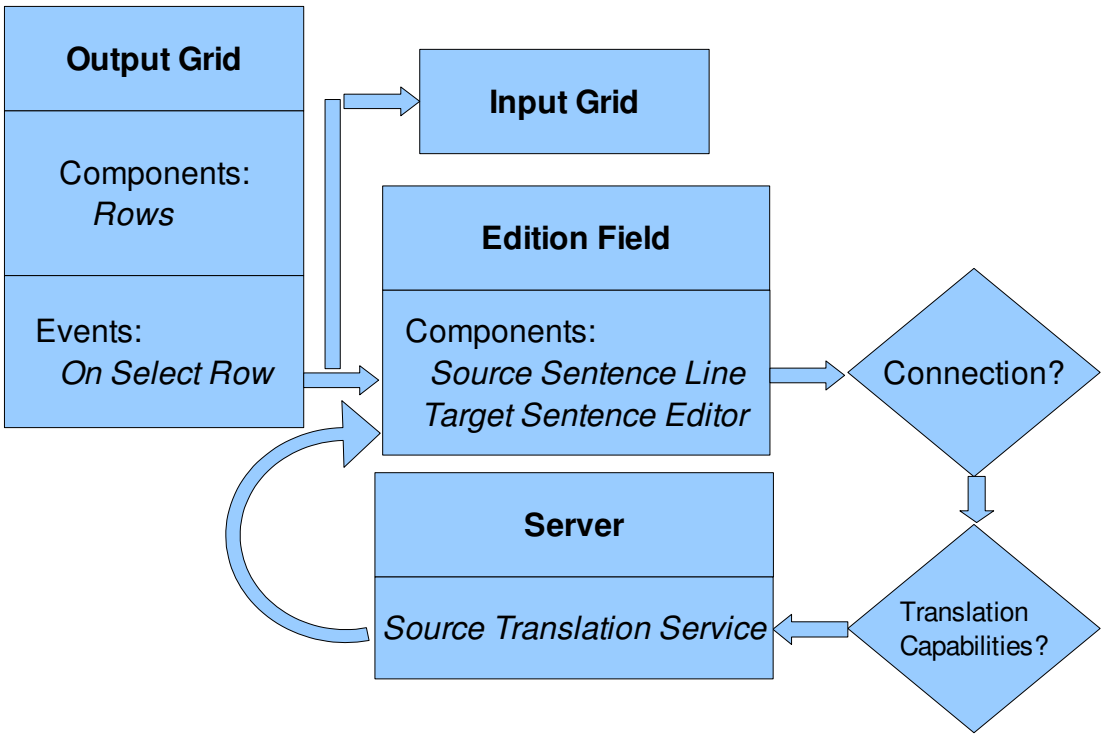
After a text file is opened, it is read line by line. Every line having the sentence separator is divided into two parts; if no separator is found, the line is considered as a complete sentence. A row with two cells is added to the grid and the sentences are introduce one to every column. If there is only one sentence, it is introduced in the first column. If several files were selected and designed in the explorer to be opened, they are sorted and opened in the resultant order. The lines of the files are introduced in the same order of the sorted file list.

There is only one event associated to the input grid:

- *On Select Row*: When a row from the input grid is selected, the line being in the edition field, if there is one, is transferred to the last position of the output grid, both source sentence and target sentence, and all rows being above the selected row are transferred to the output grid after the processed sentence, in the same order they had. The source sentence of the selected row is transferred to the first line of the edition field. At the same time, if there is a connection to a server, an if this server is enabled to translate text, it is sent a request to translate the sentence. If there is a positive response, the sentence obtained in the process is presented in the second line of the edition field, a text editor. If no response is obtained or there is not a proper connection, the second line stay empty and the edition capacities are disabled.

4.2.3.2 Output Grid

The output grid is also a Tree View. Its structure is rather the same as the one of the input grid. In the first column of the output grid the source sentence is presented, in the second one the translation obtained in the computer assisted translation is presented. The sentences presented in the output grid can be saved to a text file.

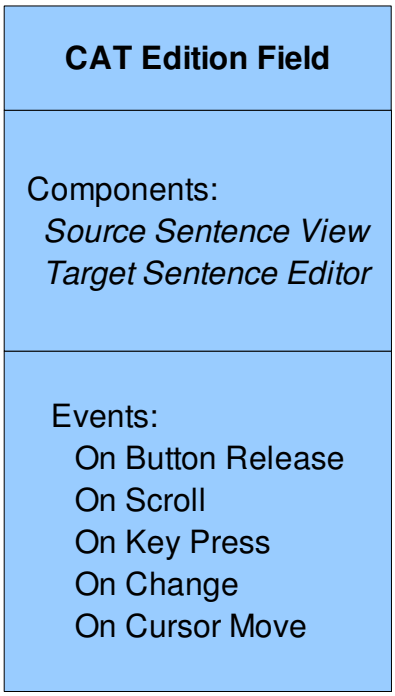


There is only one event associated to the output grid:

- *On Select Row*: When a row from the output grid is selected, if there is one line in the edition field, it is transferred to the first position of the input grid, both source sentence and target sentence, and all rows being below the selected row are transferred to the input grid before the processed sentence, in the same order they had when the file was

opened. The source sentence of the selected row is transferred to the first line of the edition field. At the same time, if a connection to a server is enabled, an if this server has the capability to translate text, it is sent a request to translate the sentence. If there is a response, the sentence sent by the server is presented in the second line of the edition field. If no response is obtained or there is not a connection, or if the translation service is not included between its capabilities, the second line stays empty and the field edition capabilities are disabled.

4.2.3.3 CAT Edition Field



The CAT edition field is the most important section of the program, is the place where computer assisted translation takes place. The edition field consists of

two text views. The first one aimed at the source phrase and the second to the target phrase. The first text view has only a purpose, present on the screen a sentence. But it has no other function, because the text inside the text view can't be modified. In order to make the edition field easily distinguishable from the input and output grids, it has been assigned a different colour. This facilitates the work of the human corrector.

The second text view is a text editor, with some special capabilities intended to capacitate the user to perform computer assisted translation. To implement CAT operation, some events have been enabled on this text editor. The implemented events are the following:

- *On Button Release*: This event is raised when a mouse button is released. The Button Release event manages two possible situations, namely, cursor insertion and end of text selection. The Button Press event can't manage the latter case, for that reason the Button Release signal was selected for these purposes.

  In the first case, when the user inserts the cursor at some location, it must be understood that the text is accepted until that point, and in that location the user has found an error or the user would prefer another variant. That point is then considered a prefix bound, this is, the settled fragment. The prefix detached from the sentence is sent to the server, if this has completion capabilities. If the server returns a suffix, a new sentence is assembled using the settled prefix and the new suffix and it is presented as suggestion to the user.

  If the server allows char level operations, the setting prefix request is

sent independently to the position where the cursor was inserted. Instead, if the server allows only word level operations, the setting prefix request is sent only when the cursor is inserted after a space.

The second case, when the user selects some range of text dragging the mouse on it, is treated rather differently. The text editor used in CAT do not have text formatting facilities. Then, there is no reason to select text different to the user desire of substitute the selected text range for another word or simply delete it. In this case the section selected is deleted and the *free edition state* is enabled to let the user, if he desires, write a new word. Text can be selected using the left mouse button "*double click*" too; in this case a word is selected when the "*double click*" is performed over it. The word selected with this method is deleted too.

- *On Scroll Event*: The scroll signal is raised when the mouse wheel is rotated while the mouse pointer is over the text editor area. The scroll signal is managed only after a prefix has been settled. This happens only if the cursor has been already inserted at any point inside the text. The first scroll after an insertion is used to ask to the server for *n* possible completions of the same prefix. Once the completions have been obtained, the first one is appended to the prefix and presented in the editor. Further scrolling don't generate more consultation to the server. In this case, the suffixes are appended in order to the prefix and proposed to the user. Up scrolling is used to present previous suffix, down scrolling is used to present next suffix. When the available suffixes have been presented, the last or the first is presented, depending of the scroll direction.

- *On Key Press Event*: Key hit can be exploited to offer to the user some edition options, intended to increase edition efficiency. It can be interpreted as the beginning of text edition too, but this case it is left to the *On Change Event*, when the key signal process is ended. The Key combinations managed in the Key Press signal are the following:

  "*Control + Page Up*": this key combination has been enabled to move and translate the next sentence in the queue of the input grid.

  "*Control + Page Down*": this key combination has been enabled to move and translate the previous sentence in the queue of the output grid.

  "*Control + Up Arrow*": this key combination has been enabled to present the next one of *N* completions, as explained in the *Scroll Event*.

  "*Control + Down Arrow*": this key combination has been enabled to present the previous one of *N* completions, as explained in the *Scroll Event*.

  "*Control + Delete*": this key combination has been enabled to delete the next word after the cursor, or if the cursor is inside a word, it deletes the word at the cursor position. When this happens a state of "free edition" is activated. This is explained next, in the case when the F5 key is pressed. The state of "free edition" can be ended hitting "enter" key.

  "*Control + Backspace*": this key combination has been enabled to delete the previous word to the cursor position. If the cursor is inside a word, it

deletes the word at cursor. This key combination activates the “free edition” state too.

“*F5*”: The F5 function key has been enabled to toggle between a state of *free edition* and a state of CAT edition. In the state of *free edition* no use is done of the server CAT capabilities. This is included to let the user to do operations of deletion, insertion an substitution. The assistant algorithm is inhibited to do any suggestion at all. Too much support can be embarrassing and distracting and can hinder the efficiency of the edition process. For that reason it is useful to let the user deactivate the computer assisted translation temporarily. When the free edition is activated the back colour of the editor is changed to let know the user, at every moment, the editor state. The state of free edition is activated with the hit of other key combinations like “*Control + Delete*” or “*Control + Backspace*”, or when a range of text is selected. The state of free edition can be finished up in two ways. The first one entails hitting once more *F5* key. In this case no completion is provided. The second one consists in pressing the *enter* key. In this last case, the text range bound by the cursor is separated as a prefix and CAT completion is forced. In any case the editor's back colour is restored to the previous value. Simultaneously CAT capabilities are restored

“*Enter*”: The *enter* key has been enabled to put an end to the state of *free edition*. When the enter key is pressed with the *free edition* state activated, the cursor is advanced from its current location to the end of the next word. Up to that point, a settled prefix is established and the server is consulted for completion. The resulting suffix and the settled prefix are assembled and presented to the user in the target sentence

editor. The editor's back colour is restored to the previous state and CAT capabilities are restored.

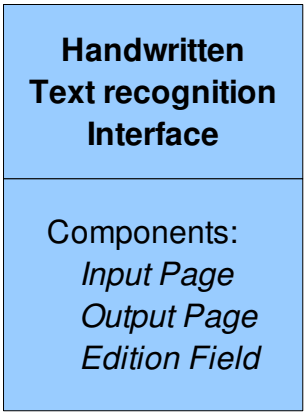
- *On Key Release Event*: Key Release signal is use to catch user text selection using keyboard. Text selection is performed using "*Shift + Right Arrow*", "*Shift + Left Arrow*", to select a character at a time, and "*Shift + Control + Right Arrow*", " *Shift + Control Left Arrow*", to select word a word. When selection is ended, the selected text is deleted and the *free edition state* is activated.

- *On Change Event*: The *Change Event* is raised when a change has been made to the text, that is, when a character has been inserted at some point or a character has been deleted backward or forward at cursor location. At every character insertion or deletion, the text, from the beginning up to the cursor position, including the new character, is established a settled prefix. The server is requested for completion and and the resulting suffix added to the settled prefix is proposed to the user as source sentence translation.

4.2.4 Handwritten Text Recognition interface

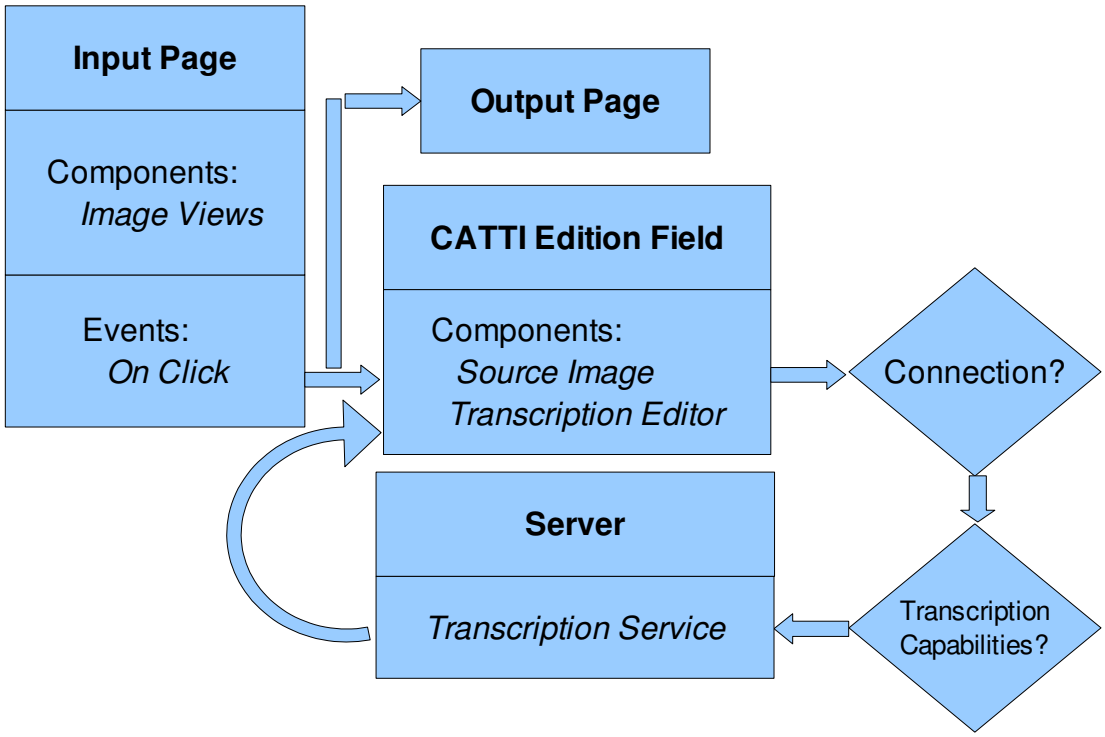
The HTR interface is made up of three components, namely, an input page, an edition field and an output page. This interface is used to carry out text transcription activities applied on images taken from graphic files. The HTR Interface is intended to implement it, using the connected server capabilities. For this purpose we use an input page, and an output page. These pages consist of sets of images, optionally intercalated by text boxes where transcription results are presented. The pages can be configured to show either only the images, either only the image's transcriptions or both, the images and the transcriptions. The two pages can contain any number of images.

When a set of image files are opened, they are sorted by file name, in increasing alphabetical order and disposed one down the other piled inside the input page. The user can select an image to be processed. The selected image is transcribed and the result presented in the edition field. Computer assisted corrections can be effectuated, using the offered server capacities. When a new image is selected to perform HTR process, the former image, and its corrected and validated transcriptions, are sent to the output page.



The CATTI edition field is built up of two components, an image view used to present the image intended to be transcribed in the edition process, and a text view with edition services, aimed to let the user to interact with the CATTI server to perform computer aided corrections and to make local free edition operations.

4.2.4.1 Input Page



The Input Page is a vertical GTK container (GtkVBox) containing a set of image views and label renderers intercalated. The graphic files opened by the user are situated one in each image view. The user can open any number of files, because image and label widgets are created dynamically. The images are scaled to a size that fits the screen used; this is done with the purpose of take advantage of the screen space available trying to avoid the use of scroll

bars that distracts and spends the user's time. In any way the vertical box has scroll bars, vertical and horizontal, in the case they be needed when the user change the window's size.

Th input page has an important event, needed to select the image to be processed:

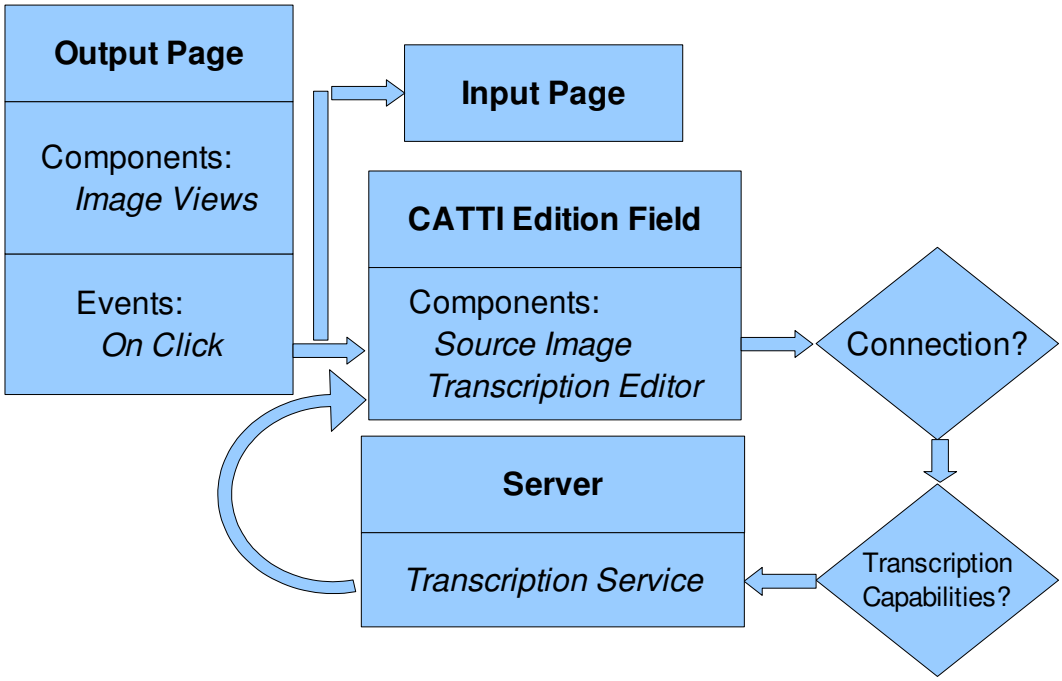
- *On Click*: The click event is raised when the user hits the left mouse button when the mouse pointer is on an image of the input page. When the labels are visible, the click event is raised when the mouse is clicked over one of them too. When an image is selected, the image and the associated label are sent to the CATTI editor. If the server has transcription capabilities, a transcription request is sent and if an answer is obtained, it is presented in the text editor, ready to be edited. The images being above the selected image are sent in the same order they have to the output page. The remaining images being under the selected image are kept in the input page. If no answer is obtained from the server, the edition services are disabled and the target line remains empty.

4.2.4.2 Output Page

The output page is a vertical box with scroll bars. When the user opens graphic files, the output page remains empty. It receives the image views and the labels extracted from the input page when an image or label is clicked, or when

a transcription is validated. When the “*save*” or “*save as*” menu items are selected the label's contents of the output page are saved in a text file.

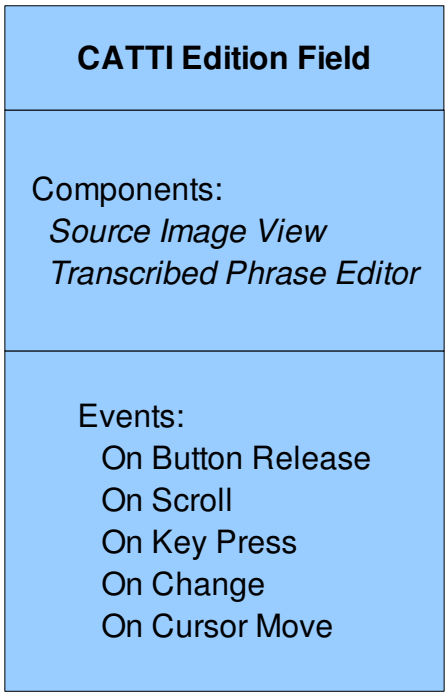
The output page has an event:



- *On Click*: The click event is raised when the user hit the left mouse button when the mouse pointer is on an image or a label of the output page. When an image is selected, the image and its corresponding label are sent to the CATTI editor. If the server has transcription capabilities, a transcription request is sent and if an answer is obtained, it is presented in the text editor, where it can be edited. The images being under the selected image are sent in the same order they have to the beginning of the input page. The remaining images being above the selected image are kept in the output page. If no answer is obtained from the server, the edition services are disabled and the target line remains empty.

4.2.4.3 CATTI Edition Field

The CATTI edition field is the place where the user performs computer assisted transcription aided by the system. The edition field consists of a image view and a text view. The first one intended to contain the source image, the second one intended to contained the target sentence, the phrase recognized in the image by the HTR system. The image view has the purpose to present on the screen the image intended to be processed.



The text view is a text editor. User can write on it, and some special facilities have been added to allow the user to perform computer assisted transcription. To implement CATTI operation some events have been enabled on this text editor. The implemented events are the following:

- *On Button Release*: This event is raised when a mouse button is released when the pointer aims to any point of the transcribed sentence.

The Button Release event manages two following possible situations: cursor insertion and end of text selection.

The cursor insertion is used to define the prefix bound in the CATTI edition operations. When the user inserts the cursor at some point, text from the beginning of the transcription up to the insertion point is defined as prefix. That point is then considered a prefix bound and it delimits the settled fragment. The prefix removed from the sentence is sent to the server, if this has completion capabilities. If the server returns a suffix, a new sentence is formed using the settled prefix and the new suffix and it is presented to the user as new image's transcription.

If the server allows char level operations, the setting prefix request is sent independently of the position where the cursor was inserted. Instead, if the server allows only word level operations, the setting prefix request is sent only when the cursor is inserted after a space.

The second operation, when the user selects a range of text dragging the mouse on it, is treated in a different way. In this case the text selected is deleted and the *free edition state* is enabled to let the user, if he desires, to write new text. Double click action with the mouse left button can be used to selects text too; in this case a word is selected when the "*double click*" is performed over it. The word selected with this method is deleted too.

- *On Scroll Event*: The scroll event is activated when the mouse wheel is rotated while the mouse pointer is over the text editor area. The scroll

signal is available only after a prefix has been settled. This happens only if the cursor has been already inserted at any point inside the text, delimiting a prefix. The first scroll, after an insertion, is used to ask to the server for *n* prefix possible completions. Once the completions have been sent back from the server, the first one is added to the prefix and rendered in the text editor. Further scrolling don't generate more requests to the server. In this case, the suffixes are toggled and added in order to the prefix and presented to the user. Up scrolling is used to present previous suffix, down scrolling is used to present next suffix. When the available suffixes have been proposed, the last or the first is presented, depending of the scroll direction.

- *On Key Press Event*: Key press event is used to manage some useful key strokes and key combinations. Some edition options are offered, fired by special keys. The Key combinations managed in the Key Press signal are the following:

  "*Control + Page Up*": The key combination "Ctrl + PgUp" is used to select the next image to be processed from the input page. The image is moved to the edition field. A transcription request is sent to the server. The transcribed sentence obtained is presented in the CATTI text editor. The editor edition options are enabled.

  "*Control + Page Down*": The key combination "Ctrl + PgDn" is used to select the previous image from the output page, in order to be processed. The image is moved to the edition field. A transcription request is sent to the server. The transcribed sentence obtained is

presented in the CATTI text editor. The editor edition options are enabled..

"*Control + Up Arrow*": This key combination has been enabled to present the next one of the *n* completions obtained, when a prefix has been settled. The first stroke of the Ctrl + Up (or Ctrl + Dn) combination is used to request *n* completions, the most probables suffixes obtained from the server engine Word Graph. The next stroke of the Ctrl + Up combination is used to toggle to the next suffix, to obtain a new sentence. When the last suffix is used, a further stroke causes to be used the first suffix again.

"*Control + Down Arrow*": This key combination has been enabled to present the previous one of the *n* completions obtained when a prefix has been settled. The first stroke of the Ctrl + Dn (or Ctrl + Up) combination is used to request *n* completions, the most probables suffixes obtained from the server engine Word Graph. The next stroke of the Ctrl + Dn combination is used to toggle to the previous suffix, to obtain a new sentence. When the first suffix is used, a further stroke causes to be used the last suffix.

"*Control + Delete*": This key combination is used to delete the next word after the cursor position, or if the cursor is inside a word, it deletes the word at the cursor position. When this happens a state of "free edition" is activated. The state of "free edition" can be ended hitting "enter" key. The prefix settled in this case is not bounded by the cursor position, but the limit is extended to cover the next word after de cursor position.

“*Control + Backspace*”: This key combination is used to delete the previous word to the cursor position, or if the cursor is inside a word, it deletes the word at the cursor position. When this happens a state of “free edition” is activated. The state of “free edition” can be ended hitting “enter” key. The prefix settled in this case is not bounded by the cursor position, but the limit is extended to cover the next word after de cursor position.

“*F5*”: The F5 function key is used to toggle between a state of *free edition* and a state of CATTI edition. In the state of *free edition* no use is done of the server CATTI capabilities. This is included to let the user to do operations of deletion, insertion an substitution. The assistant service is prevented to do any suggestion at all. The *F5* stroke causes to deactivate the computer assisted transcription temporarily. When the free edition is activated the back colour of the editor is changed to let know the user, at every moment, the editor state. The state of free edition is activated too, with the hit of other key combinations like “*Control + Delete*” or “*Control + Backspace*”, or when a range of text is selected. The state of free edition can be ended in two ways. The first one entails hitting once more *F5* key. In this case no completion is provided. The second one consists in pressing the *enter* key. In this last case, a new prefix is settled but not bounded by the cursor position, but the settled prefix is extended to include the next word to the cursor position. Then a new request for completion is done to the server. The editor's back colour is restored to the previous value. Simultaneously CATTI capabilities are re-established.

“*Return*”: The *return* key is used to put an end to the state of *free edition*.

When the *return* key is pressed with the *free edition* state activated, the cursor is advanced from its current location to the end of the next word. Up to that point, a settled prefix is established and the server is consulted for completion. The resulting suffix and the settled prefix are appended and the resulting sentence is proposed to the user in the target sentence editor. The editor's back colour is restored to the previous state and CATTI capabilities are re-established.

- *On Key Release Event*: Key Release signal is use to catch user text selection using keyboard. Text selection is performed using "*Shift + Right Arrow*", "*Shift + Left Arrow*", to select a character at a time, and "*Shift + Control + Right Arrow*", " *Shift + Control Left Arrow*", to select word a word. When selection is ended, the selected text is deleted and the *free edition state* is activated.

- *On Change Event*: The *Change Event* is activated when a change has been made to the text, that is, when a character has been inserted at some point or a character has been deleted backward or forward at cursor position. At every character insertion or deletion, the text, from the beginning up to the cursor position, including the new character, is established a settled prefix. The server is requested for completion and and the resulting suffix added to the settled prefix is proposed to the user as image transcription.

4.2.5 Menu bar

The menu bar contains some commands useful to perform important basic operations. The menu bar layout is designed to include common standard

commands along with specific commands. Some menu items can be executed using shorcuts and tool bar commands too, in order to implement command redundancy, useful to lend an agile service to the user. The more important commands are described in the following sections.

4.2.5.1 File Menu Item

The file menu includes basic file operations.

- *Clear Command*. The *Clear* command closes the files currently used, text files or image files. The interface currently used is cleaned. The MT interface input and output grids are emptied. The HTR images views are cleared. The edition field is cleaned and the edition services are disabled. An empty MT interface is presented on screen.

- *Open File Command*. When the open file menu command is selected a standard open file dialogue is displayed. Multiple files can be selected. If the files selected are text files, the MT interface is displayed and the text is presented in the input grid. If the files selected are image files, the HTR interface is displayed and the images are opened and displayed in the input page. The operation performed by this command is the same implemented in the directory explorer.

- *Save File Command*. The *Save File* menu command is used to save the results of the CAT / CATTI edition. When this command is triggered for the first time in the process of an edition task, a standard dialogue box is opened and it waits for the user to select a directory and a name for the file. The file is saved in text format. A new call to this command on

the same task does not prompt the user to input a file name, but the saving action is performed immediately, using the name already used, overwriting the existing file.

- *Save As... File Command*. The *Save As* command is used to duplicate a file using a different name on in another directory. The *Save As* dialogue box prompt the user or the new name or new directory. The selected name is used or further saving command activations.

- *Exit Command*. This command closes all files, deallocate memory used and closes the program.

4.2.5.2 View Menu Item

The view menu groups the commands affecting the component visibility and appearance. Some of them are:

- *Decrease Text Size Command*. This command is used to decrease the edition field components text size. The de*crease Text Size* command lets the user to adjust the text size to meet his needs. Every call to this command decreases the text size of the source text view and of the target text editor in one typographical point.

- *Increase Text Size Command*. This command is used to increase the edition field components text size. The *Increase Text Size* command lets the user to adjust the text size to meet his needs. Every call to this command increases the text size of the source text view and of the target text editor in one typographical point.

- *Normal Text Size Command*. This command is used to set the edition field components text size to a standard size. The Normal *Text Size* command lets the user to adjust the text size to a medium size. This command undo the effect of the use of the decrease and increase text size commands.

4.2.5.3 Connection Menu Item

The connection menu item groups the commands affecting the server connection state.

- *Connect Command.* The *connect* command opens the server connect dialogue box, prompting the user to input host name and port number. On accepting the connection is requested to the server. The connect dialogue box can disconnect a previous connection too.

- *Connect Info Box Command*. The *Connect Info Box* Command toggles the visibility of the connect information box.

4.2.6 Tool bar

The tool bar groups a set of commands commonly used, easily triggered by small graphical buttons. A tool bar provides agility to the use of a program, and adds redundancy to the command system. The commands implemented in the program tool bar are the following.

- *Clear Interface Tool Command.* The *Clear Interface Tool* Command resets the interface's components. The files used are closed, the input grid and the output grid are cleared, the saving file name is reset, and a new translation interfaces is presented ready for another task.

- *Open Files Tool Command*. The *Open Files Tool* command displays the open file dialogue box. Multiple files can be selected in the dialogue navigator. If the files selected are text files, the machine translation interface is displayed and the text is presented in the input grid. If the files selected are graphic files, the handwritten text recognition interface is displayed and the images are opened and displayed in the input page. The operation performed by this command is the same implemented in the directory explorer and in the open file menu command.

- *Save Job Tool Command*. The *Save Job* tool command is used to save the results of the edition process. When this command is activated for the first time in the process of an edition task, a standard save file dialogue box is opened and it waits for the user to select a directory and a name for the file. The file is saved in text format. A further call to this command on the same task does not prompt the user to input a file name, but the saving action is performed directly, using the name already used, overwriting the existing file.

- *Exit Program Tool Command*. The *Exit Program* tool command closes all files, deallocate memory used and closes the program.

- *Hide/Show Explorer Tool Command*. The *Hide/Show Explorer Tool* Command toggles the visibility of the explorer child window.

- *Connect Info Box Tool Command*. The *Connect Info Box Tool* Command toggles the visibility of the connect information box.

- *Column Set-up Tool Command*. The *Column Set-up Tool* Command affects the appearance of the machine translation input and output grids, and the appearance of the handwritten text recognition input and output pages. Clicking this button permits to show either the source sentences, either the target sentences or both in the MT interface. When the interface displayed is the HTR interface, the toggling of the *Column Set-up Tool* Command lets to show either the source images, either the transcribed sentences or both.

- *Increase Text Size Tool Command*. This command is used to increase the edition field components text size. The *Increase Text Size* command lets the user to adjust the text size to meet his needs. Every call to this command increases the text size of the source text view and of the target text editor in one typographical point.

- *Decrease Text Size Tool Command*. This command is used to decrease the edition field components text size. The de*crease Text Size* command lets the user to adjust the text size to meet his needs. Every call to this command decreases the text size of the source text view and of the target text editor in one typographical point.

- *Normal Text Size Tool Command*. This command is used to set the

edition field components text size to a standard size. The Normal *Text Size* command lets the user to adjust the text size to a medium size. This command undo the effect of the use of the decrease and increase text size commands.

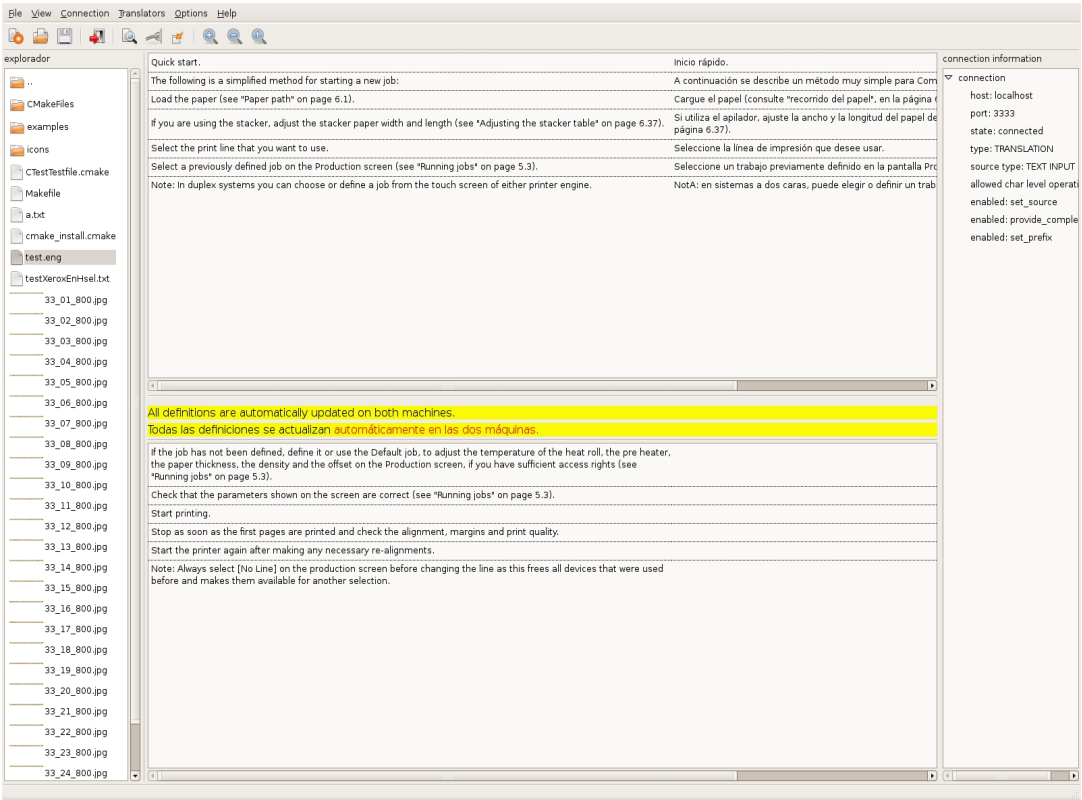
5 Result Overview

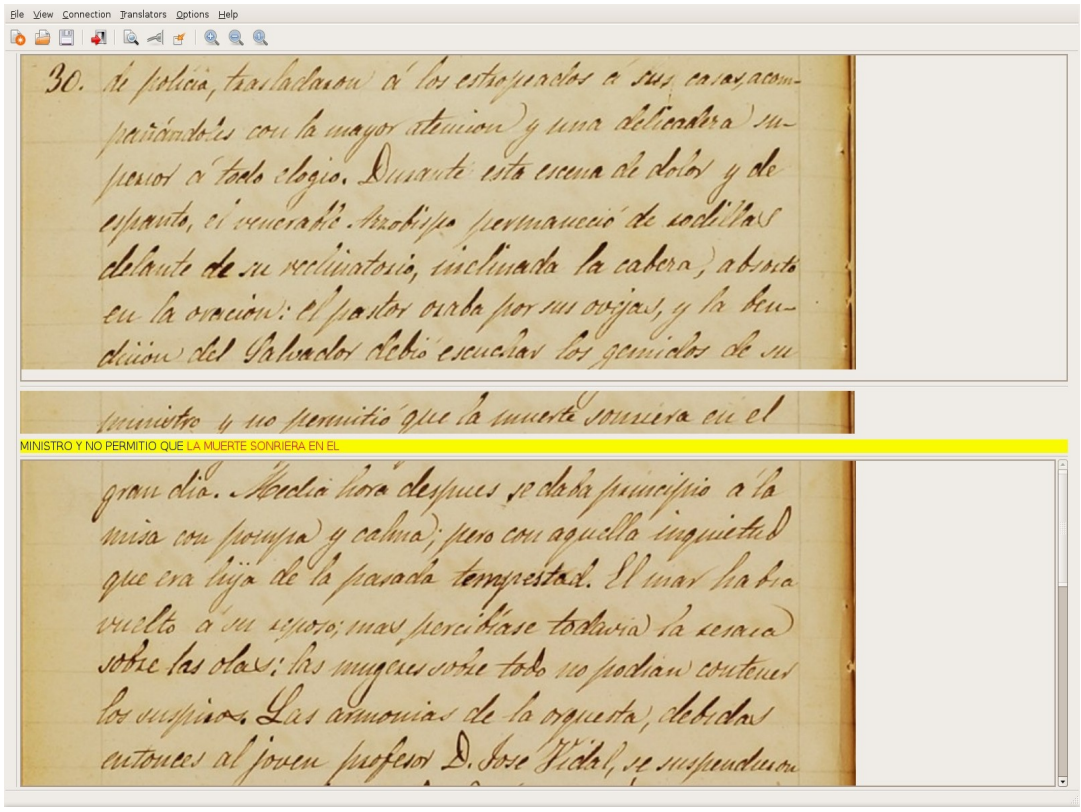
5.1 Program Overview

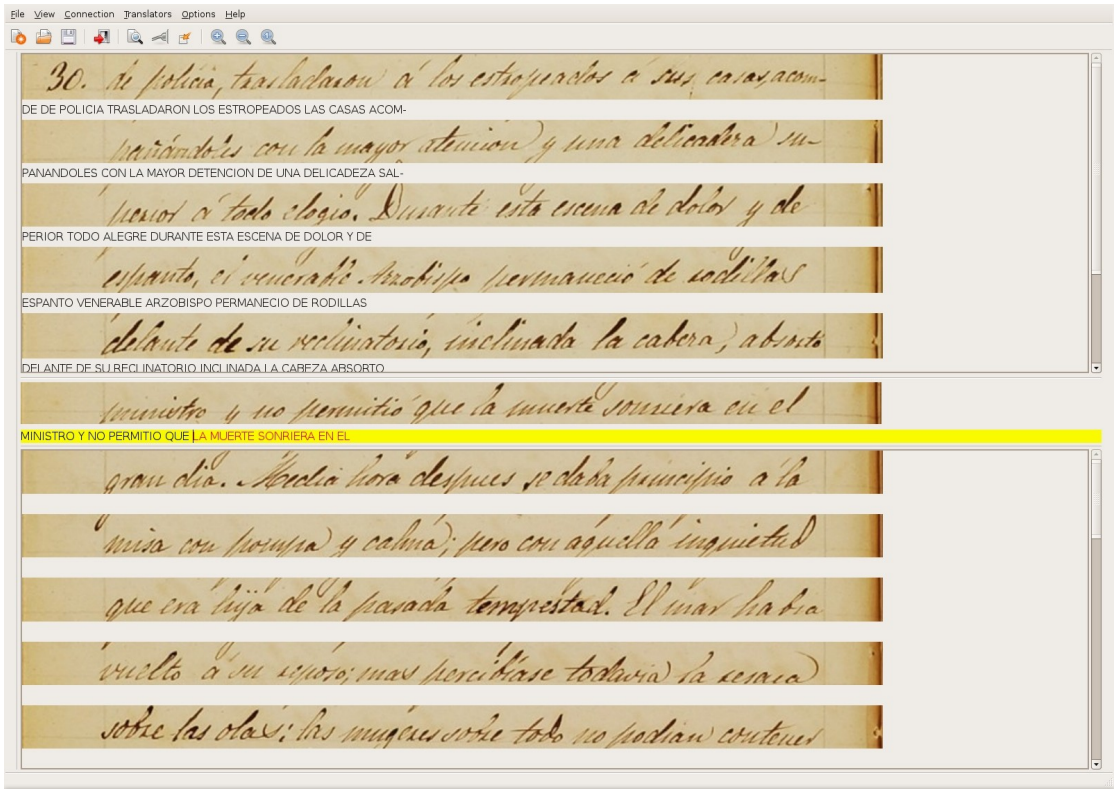
Here there are presented some program images. The first image corresponds to the MT interface, showing the explorer and the connection info box. The other three image correspond to the HTR interface in different visual settings.

The physical implementation was thought to meet some general criteria of software design. Between them it can be mentioned the following:

Usability, The graphic interface must provide easy to use background. For example, If the user desires to perform assisted translation or assisted text recognition it must be easy and intuitive to select the correct interface. Using the principle that the task that the machine can do automatically the user must not be asked to do, to select an interface it is enough to open the correct file. When the user opens text files the MT interface is displayed, and if the user opens image files the HTR interface is displayed.



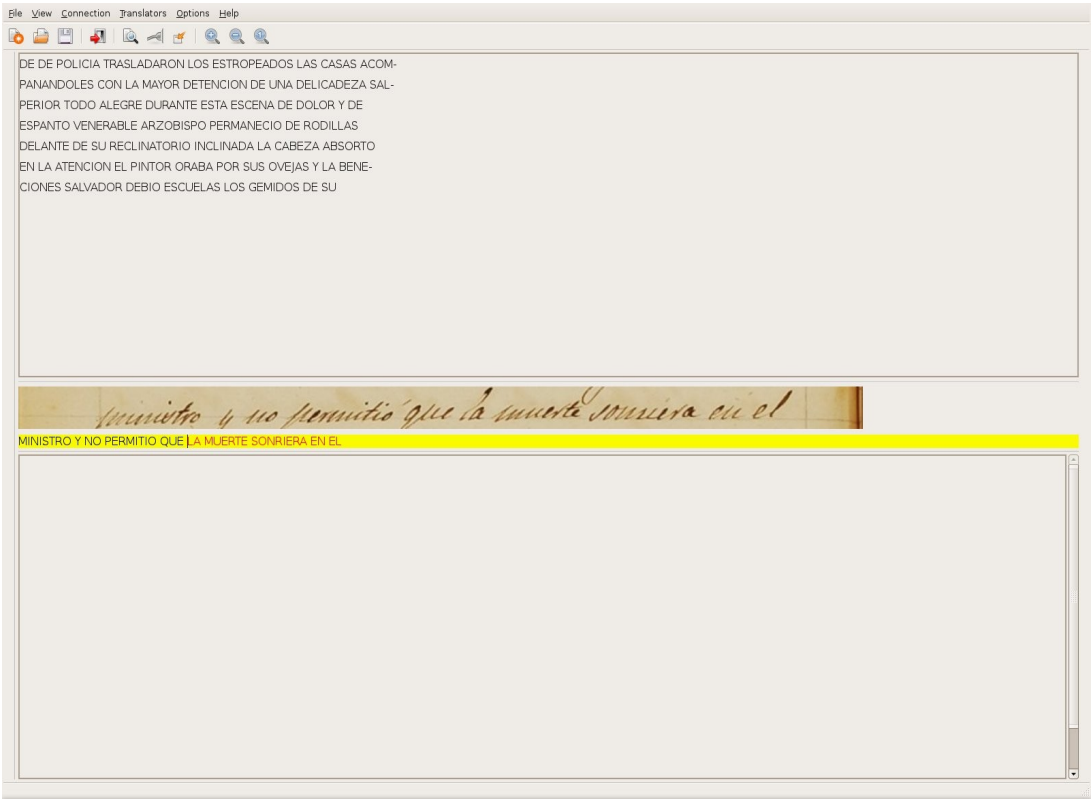

File View Connection Translators Options Help
MINISTRO Y NO PERMITIO QUE LA MUERTE SONRIERA EN EL


File View Connection Translators Options Help
DE DE POLICIA TRASLADARON LOS ESTROPEADOS LAS CASAS ACOM-
PANANDOLES CON LA MAYOR DETENCION DE UNA DELICADEZA SAL-
PERIOR TODO ALEGRE DURANTE ESTA ESCENA DE DOLOR Y DE
ESPANTO VENERABLE ARZOBISPO PERMANECIO DE RODILLAS
DELANTE DE SU RECLINATORIO INCLINADA LA CABEZA ABSORTO
MINISTRO Y NO PERMITIO QUE LA MUERTE SONRIERA EN EL

Redundancy, operations and commands can be done using multiple ways. An operation must have several possible activations: menu commands, tool bar buttons, shorcuts, etc.

Easy file operations, open files, close files, saving results, must be operation easy to perform. The program use standard dialogues to perform this tasks.

Use of software standards. The use of common and universal components. Menus, tool bars, explorers, open file dialogues, etc.



5.2 Evaluation of CAT systems

Following, some results of CAT systems valuations, using well known corpora. Measures used:

- Translation Word Error Rate (TWER): Minimum number of word insertions, deletions and substitutions needed to edit the system output into a (single) target reference .

- Word-Stroke Ratio (WSR): Number of user interactions that are necessary to achieve the reference target divided by the number of running words. In each interaction only one wrong word is changed.
- Translation Character Error Rate (CER): Minimum number of character insertions, deletions and substitutions needed to edit the system output into a single target reference.
- Key-Stroke Ratio (KSR): Number of key-strokes that are necessary to achieve a single target reference divided by the number of running characters In each interaction only one character is changed.
- Mouse Action Ratio (MAR): Ratio between the number of mouse actions required by the user to achieve the final reference sentence and its total number of words.

Here there are evaluations of CATTI systems using keyboard operations:

	IMT results with the Xerox corpus							
Data:	**SFST (1-best)**				**SFST (5-best)**			
XRCE2	**KSR**	**CER**	**WSR**	**TWER**	**KSR**	**CER**	**WSR**	**TWER**
En – Es	17,6	30,3	27,4	43,1	15,6	25,0	NA	37,8
Es -En	21,5	35,5	31,7	51,4	18,9	28,1	NA	45,2
En – Fr	37,1	54,3	65,1	73,8	34,3	48,5	NA	69,6
Fr – En	39,4	55,3	58,5	7,9	36,7	49,5	NA	67,7
En -De	38,8	62,8	55,4	81,3	35,4	56,7	NA	77,2
De – De	36,4	61,5	55,0	78,5	32,9	55,1	NA	73,3

[Barrachina et al.]

	IMT results with the EU corpus							
Data:	**SFST (1-best)**				**SFST (5-best)**			
EU	**KSR**	**CER**	**WSR**	**TWER**	**KSR**	**CER**	**WSR**	**TWER**
En – Es	27,5	37,6	52,1	55,8	24,6	34,8	NA	51,7
Es -En	25,4	38,0	48,5	52,5	22,7	35,1	NA	48,0
En – Fr	26,2	36,0	62,2	53,9	23,5	33,4	NA	50,1
Fr – En	23,1	36,1	60,5	49,2	20,6	32,8	NA	44,4
En -De	29,4	41,2	49,6	65,5	26,8	38,1	NA	60,3
De – De	31,0	44,4	44,0	66,6	28,0	41,4	NA	61,2

[Barrachina et al.]

Evaluation of the mouse use as an additional information source:

Results with the Xerox corpus				
	Baseline		With mouse actions	
Data: XRCE2	**MAR**	**WSR**	**MAR**	**WSR**
En – Es	10,0	27,4	27,1	21,6
Es - En	13,5	31,7	31,3	23,8
En – Fr	12,6	65,1	65,0	59,2
Fr – En	13,5	58,5	58,4	51,9
En - De	13,6	55,4	54,9	48,3
De – De	15,7	55,0	54,6	47,1

[Sanchis et al.]

Results with the EU corpus				
	Baseline		With mouse actions	
Data: EU	**MAR**	**WSR**	**MAR**	**WSR**
En – Es	15,9	52,1	52,3	44,9
Es - En	13,8	48,5	48,2	40,8
En – Fr	23,5	62,2	62,3	56,4
Fr – En	14,4	60,5	60,5	53,4
En - De	15,6	49,6	49,8	43,1
De – De	14,4	44,0	43,8	36,3

[Sanchis et al.]

5.3 Evaluation of CATTI systems

- Word Error Rate (WER): Minimum number of words that need to be substituted, deleted or inserted to convert a sentence recognized by the system into the reference transcription, divided by the total number of words in this transcription.
- Word-Stroke Ratio (WSR): Number of (word level) user interactions required to achieve the reference transcription, divided by the total number of reference words .
- Effort-Reduction: Estimates the effort saved by CATTI with respect to post-editing .
- Word click rate (WCR): Number of additional mouse-clicks by word, divided by the total number of reference words .

			Cristo-Salvador	
	ODEC	IAMDB	Soft	Hard
WER (%)	25,3	25,6	33,9	38,9
WSR (%)	22,7	23,4	32,5	38,1
Effort-Reduction (%)	10,3	8,6	4,1	2,1
WSR (%)	19,8	16,6	27,8	33,6
Effort-Reduction (%)	21,7	23,4	18	13,8

6 Conclusions and Future Work

The automated performance of machine translation / handwritten text recognition systems is not perfect and errors spoil their results. Therefore, the search of intermediate solutions between MT and HTR and pure human translation / transcription is an urgent need.

In the performance of probabilistic MT / HTR systems, some correct solutions can be lost on the road; the human guidance can lead the systems to results faster and more efficient that the pure human translation and with results equally good.

The statistical models which drive MT and HTR systems actually capture a great deal of linguistic knowledge relevant to the tasks for which these models have been framed. Their power can be used in addition to human insight to find an economical high quality solution to MT / HTR problems. For that reason CAT / CATTI interactive technologies represent a real option to pure MT / HTR.

Endowing CAT / CATTI interfaces with additional user “local” edition capabilities, in the form of deletion, insertion and substitution can be useful to improve the services rendered by the system. The use of intuitive, easy to remember and small in quantity shorcuts and tools can lend agility to the process and increase the user control over the system.

Futures developments can include the addition, to the human-computer interface here implemented, of other types of human-computer interactive technologies. The Multi-modal Computer Assisted Transcription of Handwritten Text Images (MM-CATTI) is an edition system that makes use of touch screen communication devices to provide feedback about the correct prefix and edition operations; this is a natural and ergonomic system that renders effort saving and ease of use. Another system capable of save edition effort and time is the Computer Assisted Speech

Transcription (CAST), consisting in the use of the voice to validate correct prefixes.

7 Bibliography

[AERFAI] AERFAI Summer School on New Trends in Pattern Recognition for Language Technologies (AERFAISS'08). Bilbao, June 23-28, 2008 (www.iti.upv.es/~fcn/Talks/Bilbao08/interactiveMT_4p.pdf).

[Barrachina et al.] S. Barrachina, O. Bender, F. Casacuberta, J. Civera, E. Cubel, S. Khadivi, A. Lagarda H. Ney, J. Tomas, and E. Vidal.. Statistical approaches to computer-assisted translation. Computational Linguistics, 35(1), 2009.

[Civera et al.] J. Civera, J.M. Vilar, E. Cubel, A.L. Lagarda, S. Barrachina, F. Casacuberta, and E. Vidal. A novel approach to computer assisted translation based on finite-state transducers. In *Finite-State Methods and Natural Language Processing*, volume 4002 of *Lecture Notes in Artificial Intelligence*, pages 32-42. Springer, 2006. Revised papers of Proceedings of FSMNLP 2005.

[Casacuberta el al.] F. Casacuberta, J. Civera, E. Cubel, A.L. Lagarda, G. Lapalme, E. Macklovitch, and E. Vidal. Human interaction for high quality machine translation. *Communications of the ACM*, page In press, 2009.

[Lagarda et al.] Antonio L. Lagarda, Vicent Alabau, Carlos-D. Martínez-Hinarejos, Alejandro-H. Toselli, Verónica Romero, José-R. Navarro, and Enrique Vidal. Computer-assisted handwritten text transcription using speech recognition. In *V Jornadas en Tecnología del Habla (VJTH'2008)*, pages 229-232, Bilbao (Spain), Nov 2008.

[Rodríguez et al.] L. Rodríguez, F. Casacuberta, and E. Vidal. Computer assisted transcription of speech. In *Proceedings of the 3rd Iberian Conference on Pattern Recognition and Image Analysis, Volume 4477 of LNCS*, pages 241-248, Girona (Spain), June 2007.

[Romero et al. 2007] V. Romero, A. H. Toselli, L. Rodríguez, and E. VidalA. Computer Assisted Transcription for Ancient Text Images. In *International Conference on Image Analysis and Recognition (ICIAR 2007)*, volume 4633 of *LNCS*, pages 1182-1193. Springer-Verlag, Montreal (Canada), August 2007.

[Romero et al. 2008] V. Romero, A. H. Toselli, J. Civera, and E. Vidal. Improvements in the computer assisted transciption system of handwritten text images. In *Proc. of the 8th Int. Workshop on Pattern Recognition in Information Systems (PRIS 2008)*, pages 103-112, Barcelona (Spain), June 2008.

[Sanchis et al. September 2008] G. Sanchis-Trilles, M.T. González, F. Casacuberta, E. Vidal, and J. Civera. Introducing additional input information into imt systems. In *Proceedings of the 5th Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms. Lecture Notes in Computer Sciences*, volume 5237, pages 284-295. Utecht, The Netherlands, 8-10 September 2008.

[Sanchis et al. October 2008] G. Sanchis-Trilles, Daniel Ortiz-Martínez, Jorge Civera, Francisco Casacuberta, Enrique Vidal, and Hieu Hoang. Improving interactive machine translation via mouse actions. In *EMNLP 2008: conference on Empirical Methods in Natural Language Processing*, Waikiki, Honolulu, Hawaii, October 25 - 27 2008.

[Toselli et al. 2007] A. H. Toselli, V.Romero, L. Rodríguez, and E. Vidal. Computer Assisted Transcription of Handwritten Text. In *9th International Conference on Document Analysis and Recognition (ICDAR 2007)*, pages 944-948. IEEE Computer Society, Curitiba, Paraná (Brazil), September 2007.

[Toselli et al. 2008] A. H. Toselli, V. Romero, M. Pastor, and E. Vidal. Preprocessing and Feature Extraction Technics for Multimodal Interactive Transcription of Text Images. Technical report, Instituto Tecnológico de Informática, http://prhlt.iti.es, July 2008.

[Vidal et al.] E. Vidal, L. Rodríguez, F. Casacuberta, and I. García-Varea. Interactive pattern recognition. In *Proceedings of the 4th Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms, Volume 4892 of LNCS*, pages 60-71. Brno, Czech Republic, 28-30 June 2007.