Escola Politècnica Superior de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Master of Science Thesis in
Telecommunication Engineering and Management

Word prediction for a real-time reader device for blind people

Pere Palou Llobera

Supervisors: Eros Pasero
Paolo Motto Ros

September 2008

A mon pare, a ma mare i a n'Aina,

a les meves tres padrines,

i als amics, amigues, companys i companyes
amb qui he compartit aquests anys
i què m'han ajudat a créixer com a persona
en tots els aspectes.

"Tutto ha un lato positivo, guardalo e fallo vedere agli altri"

Contents

Introduction

Nowadays, printed books and magazines still represent the biggest amount of readable information. Therefore, the unavailability to access to such information is a big limitation.
This is the case for people that suffer blindness, which are about 37 million (0,6%) all over the world, estimated by the World Health Organization [1].

Blind people are able to read only if characters are translated into Braille tactile alphabet, which is ideated specially for them. Nevertheless, to translate all books into Braille alphabet becomes a complicated task due to its format[1] and, also, due to the fact that blind people represent a minority in the society.

An adopted solution to resolve the above stated issue, which is to allow blind people to read printed information, is realized in [2]. A prototype of real-time reader device (shown in **Fig. 0.1**) has been developed to optically acquire printed data, using a software based on optical character recognition (OCR) combined with artificial neural networks (ANN) algorithms.

The device is a manual scanner USB-connected to a computer, where software treats images obtained by a CMOS camera and recognize characters.

The output of the system is the **character recognized** by the software that is given to the user (blind person), on the one hand, with a Braille touching device and, on the other hand, spelled by a vocal synthesis unit.

[1] Braille alphabet format is based in six separated raised points

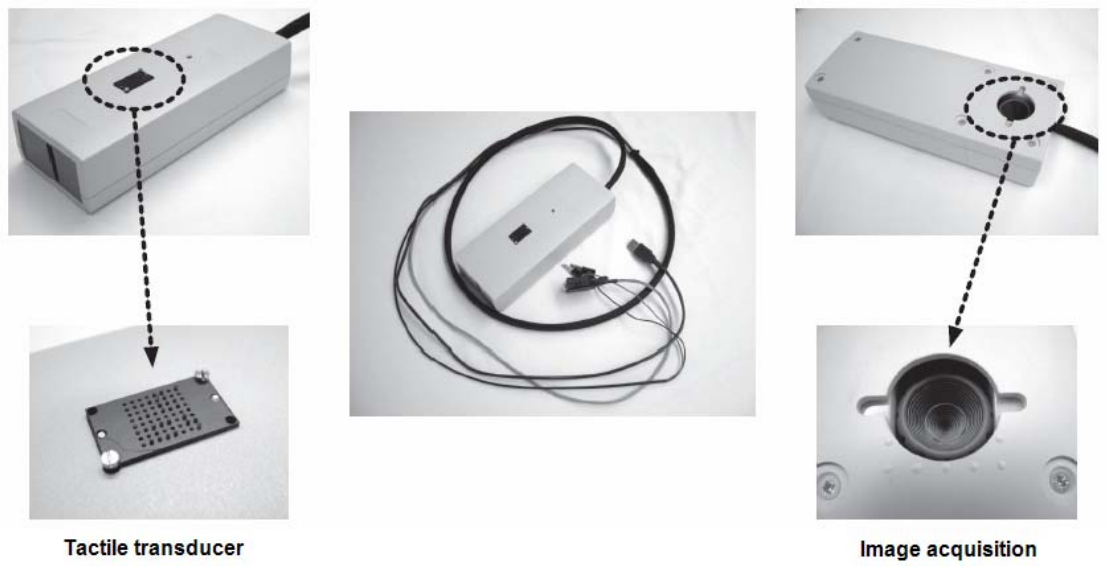
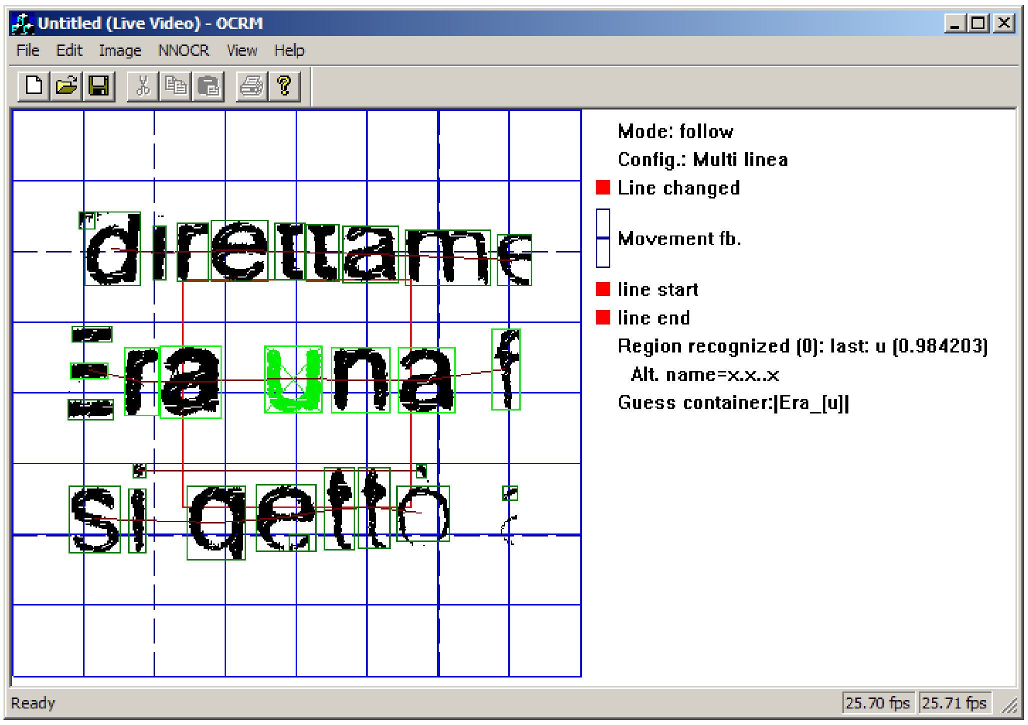


Fig. 0.1 Real-time reader device

Fig. 0.2 shows the software application, which is also able to: detect if a line has changed/started/ended, give a history of detected characters, help the user to obtain a correct acquisition of the information, and others.



Mode: follow
Config.: Multi linea
🟥 Line changed
Movement fb.
🟥 line start
🟥 line end
Region recognized (0): last: u (0.984203)
Alt. name=x.x..x
Guess container:|Era_[u]|

Fig. 0.2 OCRM software application

The transduction of a single character recognized to tactile stimulation and audio information is an important achieved task. Nevertheless, is slightly efficient for documents reading due to the fact that blind people read in a word-oriented way.

The aim of this project, taking the software developed in [2] as the starting point, is to increase the recognition reliability and robustness. The main goal of the future global system (see Chapter 1) is the ability to be the closest possible to the way that blind people read, increasing the accessibility to this group of people.

If this system can considerably help blind people to read, these people would probably get more reliability to access new technologies, due to the fact that unfortunately, nowadays, a great amount of blind people do not use computers because they can not access them.

Therefore, a way to increase the system reliability is to make it more robust. The current system based on artificial neural networks processes a character and tries to recognize it only taking into consideration its acquired image from the camera.
In consequence, the system does not take into consideration other information which would increase the system accuracy.

Other information could be the use of previous characters or some orthographic notions of the language in use, which are useful to **avoid** errors when a bad recognition has occurred.

For this reason, a character and word-level prediction systems have been implemented. On the one hand, useful to add a simultaneous way of recognition and, on the other hand, the starting point of a system able to correct characters or words in text.

This report is structured in six different chapters, which are detailed below:

The first chapter sets the project into its reference framework. In Chapter 2, a theoretical base of prediction based on statistical language models is given. Chapter 3 consists in the character-level prediction system realized. In Chapter 4 the word-level prediction system can be found. Chapter 5 contains a description of text correction methods based in prediction models developed. Finally, in the last chapter, the obtained conclusions and the future lines of investigation are exposed.

Chapter 1

General Framework

1.1. *STIPER 2* framework

The line of investigation of this project is set into *STIPER 2* (from Italian *STImolazione PERcettiva* that means perceptive stimulation) project of the INFN (Istituto Nazionale di Fisica Nucleare) [3], developed by:

- ISS (Istituto Superiore di Sanità) in Rome, Italy [4].
- Neuronica Laboratory in the Department of Electronics of the Polytechnic of Turin, Italy [5].

The goal of this global project is to study a set of instruments that can help blind people in everyday life.

The Neuronica Laboratory group is working on a part of *STIPER 2*, which the current project is focused in, which is the study and realization of a portable and autonomous system to allow blind people to read documents, thanks to the transduction of documents into tactile stimulation and audio information. As it has been introduced before, the first version was realized in [2].

At present, there are different lines of investigation in this part, to improve the present solutions.

One of this lines is the adaptation of the software developed in [2] to a new environment (DSP, PDA…) without the need of a PC, thus its portability is highly increased.

The line focused on the current project, as exposed on the introductory chapter, is to study and begin to build a set of algorithms able to predict characters based on previously recognized characters, with the aim of predicting such characters that are badly or difficulty recognized.
In addition, a first version algorithm based on word prediction is realized. The work is also focused on the application of these techniques for text correction.

1.2. Development environment

As it has been explained in the introductory chapter, the starting point software of this project is based in [2].
That software is developed on *Microsoft Windows XP SP2* operative system. The programming language used has been C++, due to the fact that:

- Is a high level programming language
- Oriented to objects programming method[2]
- Offers the possibility to build runtime working applications
- Has an available large amount of additional libraries

The software application has been implemented with *Microsoft Visual Studio 6* development environment, concretely with *Microsoft Visual C++ 6* environment.

[2] Consists in construct independent components of data structures and routines, which are declared in a *Class* (of objects), in such a way that the final program use this components, that at the same time can interact between them, in a parallel or hierarchical way.

Chapter 2

Statistical Language Modeling

As it has been stated previously, character recognition is based only on information obtained from the image acquired that has been processed by artificial neural networks algorithms.
Consequently, a new challenge would be to consider more types of information, as could be the use of previous characters recognized. This other information is needed to increase system recognition robustness.

The aim of this chapter is to introduce how natural language processing, and concretely, statistical language modeling, has been applied to achieve the above stated issue.

2.1. Natural Language Processing

In Artificial Intelligence[3], Natural language processing (NLP) is a branch that analyzes, understands and generates the languages that humans use naturally in order to interact with computers. In other words, the aim of NLP is teaching computers to understand the way humans learn and use a language.
This interaction (or learning procedure) with computers is in both written and spoken contexts using natural languages instead of computer-programming languages. A natural language (NL) is a language spoken, written, or signed by humans for general-purpose communication. English, Spanish, Catalan and Italian are examples of NL.

[3] Part of computer science which aims to create intelligence in machines

In the current project the natural language used is **Italian** as used in the system developed in [2], although it can also be trained with other natural languages.

NLP use statistical language models as a technique to deal with natural languages aspects such as syntax, semantics, etc.

The following point explains the theoretical bases of statistical language modeling applied in this project.

2.2. Statistical Language Modeling

The aim of Statistical Language Modeling (SLM) is to build a **statistical language model** able to estimate the distribution of natural language as accurate as possible.
In other words, SLM aims at capturing regularities of a language by use of statistical inference on a corpus[4] of that language.

Statistical language models predict the probability of a **text element**[5] occurrence given information about the context in which the element is expected to occur.

Language models are widely used in several natural language processing applications such as:

- Speech recognition
- Machine translation
- Information retrieval
- Optical Character Recognition (OCR), obtaining Intelligent Character Recognition (ICR)
- Handwritten recognition
- Spelling correction

2.2.1. Language models

Statistical language models most commonly used in natural language processing applications are explained below.

[4] A language corpus is a large and structured set of texts of that language.
[5] This corresponds to a single **character** or to a **word.**

A statistical language model is usually formulated as a probability distribution $p(GoE)$[6] over a group of *k* elements that attempts to reflect how frequently a GoE occurs in a text.

$$p(GoE) = p(e_1,...,e_k) \tag{2.1}$$

For example, for a language model describing spoken language, and taking the text elements as words, on the one hand, it is possible to have $p(\text{"ciao"}) \approx 0{,}01$ since one out of every hundred sentences a person speaks is *ciao*. On the other hand, having $p(\text{"la girafa nuota rapidamente"}) \approx 0{,}00$ since it is unlikely anyone says this sentence. It is important to notice that unlike in linguistics, grammar is irrelevant in language modeling.

Although there are vast amount of language models, in this paper it has been centered in n-gram models and adaptive models.

2.2.1.1. N-gram model

Nowadays, the most widely-used language models are N-gram language models. For a group of elements *GoE* composed of $e_1,...,e_k$, $p(GoE)$ can be expressed as,

$$p(GoE) = p(e_1)\cdot p(e_2|e_1)\cdot p(e_3|e_1\ e_2)\cdot\ldots\cdot p(e_k|e_1\ldots e_{k-1}) = \prod_{i=1}^{k} p(e_i|e_1\ldots e_{i-1}) \tag{2.2}$$

where $e_1...e_{i-1}$ is defined as the *history* of element e_i.

Unfortunately for artificial intelligence purposes, language is creative and being able to compute the statistics for long *histories* is a difficult task to achieve.

To overcome this limitation, a n-gram model could also be defined as non-hidden Markov model, due to the fact that it can be modeled as transition probabilities between different well-known states. Consequently, the following simplification can be assumed,

$$p(GoE)\big|_{N-GRAM} = \prod_{i=1}^{k} p(e_i|e_1^{i-1}) \approx \prod_{i=1}^{k} p(e_i|e_{i-(n-1)}^{i-1}) \tag{2.3}$$

[6] GoE: Group of text elements, can be a group of characters (GoC) or a group of words (GoW).

where *history* is reduced in the order (*n-1*) of the Markov model chain. It reduces the influence of the past elements to only the *n-1* previous elements.

Thus, besides *n* define the order of the model, the number of previous elements used in conditional probabilities is set.

If is considered the case *n* = 2, model called **bigram**, the probability of an element depends only on the identity of the immediately preceding element, giving,

$$p(GoE)\big|_{BIGRAM} \approx \prod_{i=1}^{k} p\left(e_i \middle| e_{i-1}\right)$$

(2.4)

For example, if GoE = “*Sulla scalinata che conduceva alla piazza*”,

$$\begin{aligned} p(GoE)\big|_{BIGRAM} \approx\ & p(\text{Sulla})\cdot p\left(\text{scalinata} \middle| \text{Sulla}\right)\cdot p\left(\text{che} \middle| \text{scalinata}\right)\cdot \\ & \cdot p\left(\text{conduceva} \middle| \text{che}\right)\cdot p\left(\text{alla} \middle| \text{conduceva}\right)\cdot p\left(\text{piazza} \middle| \text{alla}\right) \end{aligned}$$

Considering the case *n* = 3, model called **trigram**, which is in practice the largest *n* in wide use. In this case, the probability of an element depends on the identities of the last two preceding elements, giving,

$$p(GoE)\big|_{TRIGRAM} \approx \prod_{i=1}^{k} p\left(e_i \middle| e_{i-2}\ e_{i-1}\right)$$

(2.5)

Applying the above example in equation 2.5,

$$\begin{aligned} p(GoE)\big|_{BIGRAM} \approx\ & p(\text{Sulla})\cdot p\left(\text{scalinata} \middle| \text{Sulla}\right)\cdot \\ & \cdot p\left(\text{che} \middle| \text{Sulla scalinata}\right)\cdot p\left(\text{conduceva} \middle| \text{scalinata che}\right)\cdot \\ & \cdot p\left(\text{alla} \middle| \text{che conduceva}\right)\cdot p\left(\text{piazza} \middle| \text{conduceva alla}\right) \end{aligned}$$

In all n-gram cases, to estimate each part of equations (2.3, 2.4, 2.5) the Bayes’ theorem is applied,

$$p(A|B) = \frac{p(B|A)\cdot p(A)}{p(B)}$$

(2.6)

where the probability that an event *A* occurs given an event *B* is related to the conditional probability of *B* occurs given *A*, where

$p(A)$ is the *prior*[7] probability of event *A.*

$p(B|A)$ is the conditional probability of *B* given *A*.

$p(B)$ is the *marginal* probability of *B*, which means that is the *a priori* probability of witnessing the new evidence *B* under all possible hypotheses (A_i). It can be calculated as the sum of the product of all probabilities of any complete set of mutually exclusive hypotheses and corresponding conditional probabilities: $p(B)=\sum_i p(B|A_i)\cdot p(A_i)$.

$p(A|B)$ is the conditional probability of *B* given *A*, is also referred as the *a posteriori* probability of *B* given *A*.

Resulting,

$$p(A_1|B)=\frac{p(B|A_1)\cdot p(A_1)}{\sum_i p(B|A_i)\cdot p(A_i)} \tag{2.7}$$

The above equation (2.7) can be rewritten as

$$p\left(e_i\middle|e_{i-(n-1)}^{i-1}\right)=\frac{count\left(e_{i-(n-1)}^{i}\right)}{\sum_{e_i} count\left(e_{i-(n-1)}^{i}\right)} \tag{2.8}$$

Thus, to estimate $p\left(e_i\middle|e_{i-(n-1)}^{i-1}\right)$, the frequency with which the element e_i occurs given the last *n-1* elements, it has to be counted how often the group of n elements occurs in some text and normalize.

Taking *n* = 3, thus, for example the trigram "*che conduceva alla*",

$$p\left(e_i\middle|e_{i-2}\ e_{i-1}\right)=\frac{count\left(e_{i-2}\ e_{i-1}\ e_i\right)}{\sum_e count\left(e_{i-2}\ e_{i-1}\ e\right)}=\frac{count\left(e_{i-2}\ e_{i-1}\ e_i\right)}{count\left(e_{i-2}\ e_{i-1}\right)}=\frac{count(\text{che conduceva alla})}{count(\text{che conduceva})}$$

where $count\left(e_{i-2}\ e_{i-1}\ e_i\right)$ denote the number of times the trigram occurs in the given text.

[7] Prior means that is independent from event *B*

To develop a model, it is necessary the availability of text training data. For n-gram models, the amount of training data used is typically millions of elements. The estimate for $p\left(e_i \middle| e_{i-(n-1)}^{i-1}\right)$ given in equation (2.8) is called **maximum likelihood estimate** (MLE) of $p\left(e_i \middle| e_{i-(n-1)}^{i-1}\right)$, due to the fact that this assignment of probabilities yields the n-gram model that assigns the highest probability to the training data of all possible n-gram models.

Bigram statistical language model

As stated above, by considering *n* = 2, a bigram model is obtained, defined by equation (2.4),

$$p(GoE)\big|_{BIGRAM} \approx \prod_{i=1}^{k} p\left(e_i \middle| e_{i-1}\right)$$

(2.9)

It is important to remember that the probability of an element depends only on the identity of the immediately preceding element.

In general, for smaller *n*, there are more instances in training data (or corpus), which means a better statistical estimation and, consequently, more reliability in the model.

Trigram statistical language model

As exposed previously, considering *n* = 3, a trigram model is obtained, defined by equation (2.5),

$$p(GoE)\big|_{TRIGRAM} \approx \prod_{i=1}^{k} p\left(e_i \middle| e_{i-2}\ e_{i-1}\right)$$

(2.10)

In this case, the probability of an element depends on the identities of the last two preceding elements.

Generally, larger *n* (such as trigram model in front of bigram), more information about the context of the specific instance is obtained, thus, giving a greater discrimination of data.

It is important to emphasize that the advantages of an n-gram models are:

- Simplicity

- Easy to compute
- Well-working in predicting

2.2.1.2. Adaptive models

As it has been cited previously, natural language is creative, is highly heterogeneous, with varying genres, topics and styles.
In cross-domain adaptation, test data come from a source to which the language model has not been exposed during training (different of training data). The only useful adaptation information is in the current document itself. A quite effective technique for exploiting this information is the cache: the (continuously developing) history *h* is used to create, at runtime, a dynamic n-gram $p_{cache}\left(e\,|h\right)$, which in turn is linearly interpolated with the static model (e.g. Trigram model), resulting,

$$p_{adaptive}\left(e\,|h\right) = \lambda \cdot p_{static}\left(e\,|h\right) + (1-\lambda)\cdot p_{cache}\left(e\,|h\right)$$

(2.11)

where the λ weight is optimized on held-out data (data set not included in training corpus). In [6] and [7] different reviews of language model adaptation have been realized.

2.2.2. Smoothing techniques

For example, consider a group of two elements (a word bigram, in this case), "*più che*" that doesn't occur in the training data. Then, $p\left(\text{che}|\text{più}\right)=0$, which is clearly inaccurate as this probability should be larger than zero, due to the fact that any bigram can occur.

Smoothing is used to adjust this problem. The term smoothing describes techniques for adjusting the maximum likelihood estimate of probabilities (as in equation 2.8) to produce more accurate probabilities. The name smoothing comes from the fact that these techniques tend to make distributions more uniform, by adjusting low probabilities such as zero probabilities upward, and high probabilities downward. Not only do smoothing methods generally prevent zero probabilities, but they also attempt to improve the accuracy of the model as a whole. Whenever a probability is estimated from few counts, smoothing has the potential to improve estimation.

Additive smoothing

For a bigram model, a simple smoothing technique is to pretend each bigram occurs once more than it actually does,

$$p\left(e_i \middle| e_{i-1)}\right) = \frac{count\left(e_{i-1}\ e_i\right) + \delta}{\sum_{e_i}\left[count\left(e_{i-1}\ e_i\right) + \delta\right]}$$

(2.12)

for δ=1,

$$p\left(e_i \middle| e_{i-1)}\right) = \frac{count\left(e_{i-1}\ e_i\right) + 1}{\sum_{e_i}\left[count\left(e_{i-1}\ e_i\right) + 1\right]} = \frac{count\left(e_{i-1}\ e_i\right) + 1}{\sum_{e_i} count\left(e_{i-1}\ e_i\right) + V}$$

(2.13)

where V is the vocabulary, the set of all elements being considered. This has the desirable quality of preventing zero bigram probabilities. However, this scheme has the flaw of assigning the same probability to say, “*più che*” and “*più facile*” (assuming neither occurred in the training data), even though intuitively the first one seems more likely because the word “*che*” is much more common than “*facile”.*

Linear interpolation

To address this, another smoothing technique is to *interpolate* the bigram model with a unigram model,

$$p_{MLE}\left(e_i\right) = \frac{count\left(e_i\right)}{N_e}$$

(2.14)

a model that reflects how often each element occurs in the training data. For example, it can be taken

$$p_{\text{int}}\left(e_i \middle| e_{i-1}\right) = \lambda \cdot p_{MLE}\Big|_{bigram}\left(e_i \middle| e_{i-1}\right) + (1-\lambda) \cdot p_{MLE}\Big|_{unigram}\left(e_{i-1}\right)$$

(2.15)

getting the behavior that bigrams involving common elements are assigned higher probabilities.

Furthermore, smoothing can be applied to a trigram model, interpolating it with both bigram and unigram models,

$$p_{\text{int}}\left(e_i \middle| e_{i-2}\ e_{i-1}\right) = \lambda_0 \cdot p_{MLE}\Big|_{trigram}\left(e_i \middle| e_{i-2}\ e_{i-1}\right) + \\ + \lambda_1 \cdot p_{MLE}\Big|_{bigram}\left(e_i \middle| e_{i-1}\right) + \lambda_2 \cdot p_{MLE}\Big|_{unigram}\left(e_{i-1}\right)$$

(2.16)

Finally, it is important to notice that lots of different techniques also exists, like the Katz smoothing technique [8], the Kneser-Ney smoothing technique [9], etc.

2.2.3. Measuring model performance

Ideally, a language model performance would like to be measured by considering the decrease in error rate in the system in which the model is incorporated. However, this is often difficult to measure (as it is in our system).
The most common metric for evaluating a language model is the probability that the model assigns to test data. This metric is cross-entropy and, also, perplexity. For a smoothed n-gram model that has probabilities $p\left(e_i \middle| e_{i-(n-1)}^{i-1}\right)$, the probability of a group of elements *p(GoE)* can be calculated using equation 2.3. Then, for a test set *T* composed of the sequence of group of elements (GoE_1,…,GoE_L) the probability of the test set *p(T)* can be calculated as the product of the probabilities of all sentences in the set:

$$p(T) = \prod_{i=1}^{L} p\left(GoE_i\right) = \prod_{i=1}^{L}\left[\prod_{i=1}^{k} p\left(e_i \middle| e_1 \ldots e_{i-1}\right)\right]$$

(2.16)

The **cross-entropy** *$H_p(T)$* of a model $p\left(e_i \middle| e_{i-(n-1)}^{i-1}\right)$ on data *T* is defined as,

$$H_p(T) = -\frac{1}{E_T}\log_2\left(p(T)\right)$$

(2.17)

where *E_T* is the length of the text *T* measured in elements (words or characters). This value can be interpreted as the average number of bits needed to encode each of the E_T elements in the test data using the algorithm associated with model $p\left(e_i \middle| e_{i-(n-1)}^{i-1}\right)$.

The **perplexity** *PP$_p$(T)* of a model *p* measures the average number of successor elements which can be predicted for each element in the text. Is related to cross-entropy by the equation

$$PP_p(T) = 2^{H_p(T)}$$

(2.18)

In the worst case, the perplexity is the same as the number of words in the text.
It is important to notice that lower cross-entropies and perplexities are better.

2.3. Existing SLM packages

There are a set of common used SLM software available freely. The mostly used packages are:

- SRI Language Modeling Toolkit [10]
- CMU-Cambridge Statistical Language Modeling toolkit [11]

2.3.1. SRILM Toolkit

Although it is primarily developed for statistical tagging and segmentation, machine translation and speech recognition, SRILM is a toolkit for building and applying statistical language models. Therefore, it is useful to include it to the present system.

SRILM consists of the following components:

- A set of C++ class libraries implementing language models, supporting data structures and miscellaneous utility functions.
- A set of executable programs built on top of these libraries to perform standard tasks such as training SLM and testing them on data, tagging or segmenting text, etc.
- A collection of miscellaneous scripts facilitating minor related tasks.

In addition, is important to notice that although this package is implemented to fully work over Linux, it could also run on Windows. Nevertheless, an emulation of Linux software is needed. Despite this drawback there is a possible solution such to use *Cygwin* [12] which is a Linux-like environment for Windows. It consists of two parts:

- A DLL which acts as a Linux API emulation layer providing substantial Linux API functionality.
- A collection of tools which provide Linux look and feel.

The *Cygwin* DLL currently works with all recent versions of Windows, with the exception of Windows CE, which is the one that will possibly be used in future applications of the present system.

Note that *Cygwin* is not a way to run native Linux application on Windows. Therefore, SRILM toolkit has to be rebuilt *from source* to run it on Windows.

Setting environment variables[8] in Windows is absolutely needed to successfully compile toolkit containing files, which means getting files readable for Windows. Concretely, readable in a Visual Studio environment, but for the 2005 edition, which is a newer version that the one the system is working over (Visual Studio 6), as stated on section 1.2.

Finally, although it will be useful to apply parts of this package to newer versions of the current software, the addition of it to the present project has been discarded. Consequently, the solution adopted has been the implementation of statistical language models useful for the current software.

2.3.2. CMU Cambridge SLM Toolkit

Due to the environment similarities with SRILM toolkit, this package has been also discarded, for the same reasons exposed above.

[8] Available guide at: http://www.cs.usask.ca/~wew036/latex/env.html

Chapter 3

Character-based Prediction

The purpose of this chapter is to describe the methods used to predict characters while the recognition process is running. The recognition method built in [2] only takes into consideration the current character optically recognized allowing the user to read documents in a character by character way.

Therefore, the prediction of characters in real-time allows the system to have the availability of more information. Concretely, the availability of statistical information useful for:

- Character **recognition**, which works simultaneously with the one that already exists. This can be an starting point for:
    - A recognition system where the way of read is a group of characters instead of only one.
    - Recognize handwritten texts [13] and texts written in Asiatic languages, covering a higher range of potential users.

- Character **correction** within words.

It is important to notice that the general target of the whole project (see chapter 1) is to increase system robustness reaching a system able to allow blind people to read the most similar possible to as they read text in Braille.

3.1. Used model

The statistical language model used to is the n-gram model with n = 3. Thus, the model used to predict characters is the **statistical trigram model**.

As stated in previous chapter and extrapolated in character terms, trigram **character** modeling is defined by equation (2.10), which is,

$$p(GoC)|_{TRIGRAM} \approx \prod_{i=1}^{k} p\left(c_i \middle| c_{i-2}\ \mathrm{c}_{i-1}\right) \quad \textbf{(3.1)}$$

where,

$$p\left(c_i \middle| c_{i-2}\ \mathrm{c}_{i-1}\right) = \frac{count\left(c_{i-2}\ \mathrm{c}_{i-1}\ \mathrm{c}_i\right)}{count\left(c_{i-2}\ \mathrm{c}_{i-1}\right)} \quad \textbf{(3.2)}$$

In this case, the probability of a character depends on the identities of the last two preceding characters, ruled by the maximum likelihood estimate of $p\left(c_i \middle| c_{i-2}\ \mathrm{c}_{i-1}\right)$.

Due to the fact that our system works in real-time, trigram character model has been used as the first option given its relative simplicity of implementation.

It is important to notice that the addition of smoothing techniques in this model has been rejected. As smoothing is done to avoid zero probabilities in grams, the amount of character bigrams and trigrams not present in Italian language (e.g. such as bigram " *x s* " or trigram " *a w t* ") is quite high (see next section), consequently, if smoothing is applied to avoid negligible cases of disappearance, the probabilities of cases much more probable will considerably fall. For those reasons, smoothing on this model has been rejected.

Due to the huge length of training sets in terms of characters, an approximation can be done in not even use the additive smoothing to avoid the cases in which a bigram or a trigram with 0 occurrence is appeared, thus the amount of those bigrams/trigrams are almost ≈ 0, as it has been verified.

In addition, some type of adaptive statistical model (see section 2.2.1.2) based on this model can be applied in a near future to reach more accurate results.

3.2. Model training

Statistical language models need to be trained with data. This data can be an amount of different types of text, usually known as linguistic *corpus* (in plural: *corpora*). Models work better when trained on large amounts of data.

3.2.1. Data sets

Table 3.1 shows both constructed sets of *corpora*, which have been extracted from [14]. Its aim is to train the trigram character model.
On the one hand, the first training set is composed of text files (*.txt) with different types of text books, such as theatre comedies, dramas, novels, etc. and a sort of restaurant menus.
On the other hand, the second training set constructed is composed of text files (*.txt) of novels.

Table 3.1 Training sets specifications

Training Set	Text file	# Words	# Characters
Mixed	*La Sacra Bibbia*	765407	4545765
	La Divina Commedia (Dante Alighieri)	101493	535554
	Don Chisciotte della Mancia (Miguel de Cervantes)	367643	2169613
	Amleto (William Shakespeare)	43067	250066
	Giulio Cesare (William Shakespeare)	30979	185383
	Romeo e Giulietta (William Shakespeare)	35680	213032
	Commedie (Italo Svevo)	212632	1244932
	Eva (Giovanni Verga)	28103	174639
	Sort of restaurant menus	1961	12743
	TOTAL (in brackets, only valid characters)	1586965	9331727 (**8700555**)
Novels	*L'assassinio di Via Belpoggio* (Italo Svevo)	23827	146881
	Novelle per un anno (Luigi Pirandello)	854336	5185792
	Tutte le novele (Giovanni Verga) • *Primavera e altri racconti* • *Vita dei campi* • *Novelle rusticane* • *Per le vie* • *Drammi intimi* • *Vagabondaggio* • *I ricordi del capitano d'Arce* • *Don Candeloro e C.* • *Racconti e bozzetti*	272387	1605175
	TOTAL (in brackets, only valid characters)	1150550	6937848 (**6430718**)

In spite of the fact that in Italian alphabet letters *j, k, w, x,* and *y* do not exist, they have been considered due to the fact that, nowadays, a lot of words from other languages containing those letters, such English words (besides words of Greek and Latin origin), are adopted in the Italian vocabulary.

Consequently, it has been considered as valid characters, characters from *a* to *z* (thus, English alphabet), and space "_". Two or more consecutives spaces are neither considered valid characters.

3.2.2. Model construction

Training *corpora* is used to construct the model which assigns occurrences corresponding to each appeared:

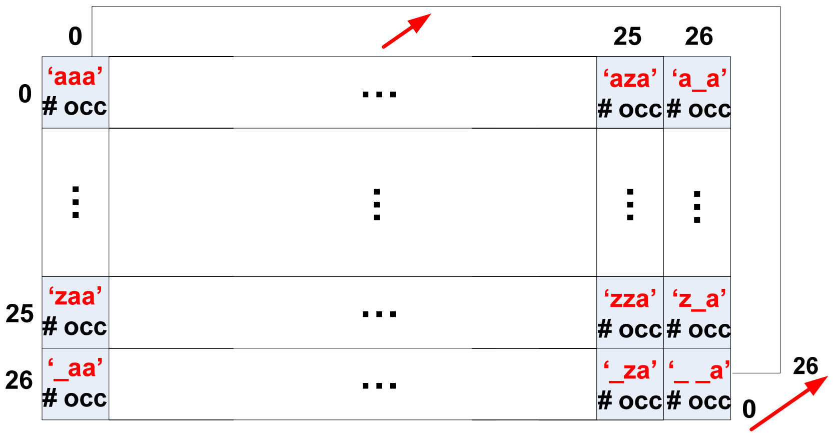
- **Unigram**. Which are 27 different character unigrams, from *a* to *z* and space "_". A one-dimensional matrix is implemented where each position corresponds to a unigram and its occurrences.

'a' # occ	…	'z' # occ	'_' # occ
0	…	25	26

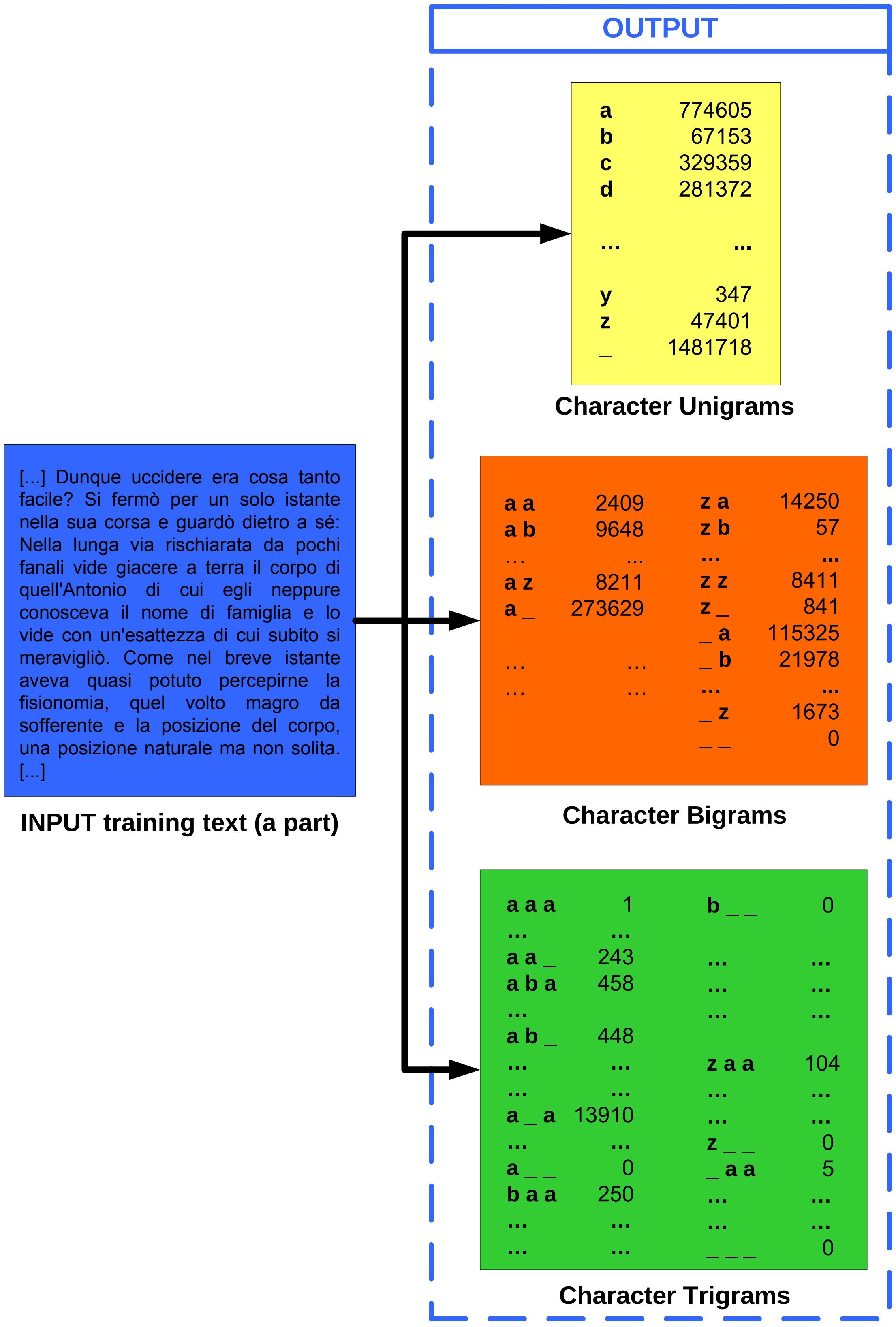
- **Bigram**. There are 27^2 = 729 possible character bigrams. In this case a bi-dimensional matrix is implemented where each position corresponds to a bigram and its occurrences.

	0		25	26
0	'aa' # occ	…	'az' # occ	'a_' # occ
	⋮	⋮	⋮	⋮
25	'za' # occ	…	'zz' # occ	'z_' # occ
26	'_a' # occ	…	'_z' # occ	'__' # occ

- **Trigram**. There are 27^3 = 19683 possible character trigrams. In this case a tri-dimensional matrix is implemented where each position corresponds to a trigram and its occurrences.



The following **Fig. 3.1** show the input of training system (an example text) which assigns occurrences to each character unigram, bigram and trigram defined below.



[...] Dunque uccidere era cosa tanto facile? Si fermò per un solo istante nella sua corsa e guardò dietro a sé: Nella lunga via rischiarata da pochi fanali vide giacere a terra il corpo di quell'Antonio di cui egli neppure conosceva il nome di famiglia e lo vide con un'esattezza di cui subito si meravigliò. Come nel breve istante aveva quasi potuto percepirne la fisionomia, quel volto magro da sofferente e la posizione del corpo, una posizione naturale ma non solita. [...]

a	774605
b	67153
c	329359
d	281372
...	**...**
y	347
z	47401
_	1481718

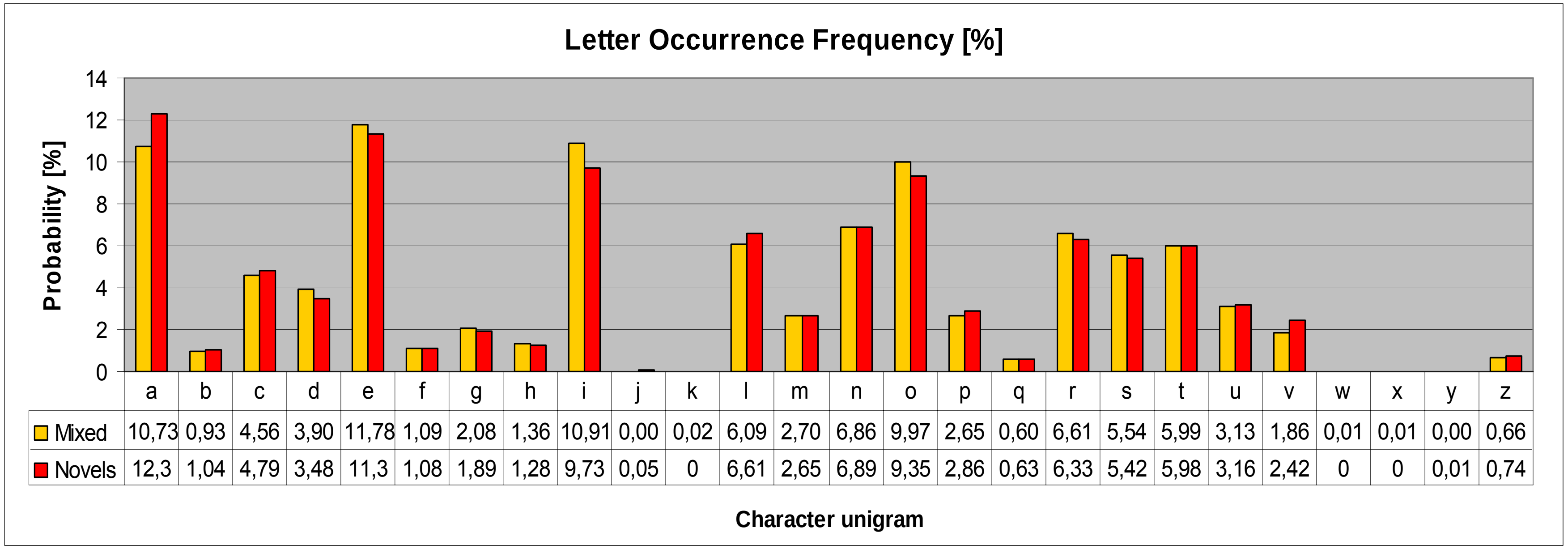
a a	2409	**z a**	14250
a b	9648	**z b**	57
…	…	**...**	**...**
a z	8211	**z z**	8411
a _	273629	**z** _	841
		_ **a**	115325
…	…	_ **b**	21978
…	…	**...**	**...**
		_ **z**	1673
		_ _	0

a a a	1	**b** _ _	0
...	**...**		
a a _	243	**...**	**...**
a b a	458	**...**	**...**
...		**...**	**...**
a b _	448		
...	**...**	**z a a**	104
...	**...**	**...**	**...**
a _ **a**	13910	**...**	**...**
...	**...**	**z** _ _	0
a _ _	0	_ **a a**	5
b a a	250	**...**	**...**
...	**...**	**...**	**...**
...	**...**	_ _ _	0

Fig. 3.1 Constructing a character n-gram list

3.2.2.1. Character unigrams

Training the trigram model is, obviously, the first step done in the system, thus, before the real time application is initialized. Therefore, while characters are being recognized, the access to characters predicted is immediate. **Fig. 3.2** shows the probability in percentage of each character unigram in both training sets specified in **Table 3.1**.



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
🟨 Mixed	10,73	0,93	4,56	3,90	11,78	1,09	2,08	1,36	10,91	0,00	0,02	6,09	2,70	6,86	9,97	2,65	0,60	6,61	5,54	5,99	3,13	1,86	0,01	0,01	0,00	0,66
🟥 Novels	12,3	1,04	4,79	3,48	11,3	1,08	1,89	1,28	9,73	0,05	0	6,61	2,65	6,89	9,35	2,86	0,63	6,33	5,42	5,98	3,16	2,42	0	0	0,01	0,74

Fig. 3.2 Character unigrams frequencies [%] for both training sets

Firstly, it can be seen that probabilities of letters *j, k, w, x,* and *y* are ≈ 0 due to the fact that the use of those characters in Italian vocabulary is almost negligible. Nevertheless, the reason of the consideration of those letters is that it makes the system scalable to other languages, due to the fact that mostly languages use them.

In addition, vowels *a, e, i* and also *o* are the characters more probable in Italian texts. Furthermore, consonants *l, n, r, s* and *t* are the more present ones.

3.2.2.2. Character bigrams

Firstly, **Fig. 3.3** is a distribution of the 30 most probable character bigrams for mixed training set. Reductions of stored bigram information can be applied knowing that:

- An 18,38% of bigrams have 0 occurrences.
- A 47,05% of bigrams (including those with 0 occurrences) have 4 magnitude orders less than the most common bigrams.

If this is taken into consideration, those bigrams can be suppressed, remaining a 99.98% of bigram occurrences. The remaining percentage of bigrams, is mostly covered by only the 30 most probable bigrams, which are the 43,66% of all 27^2 bigrams.

It can be observed that most probable bigrams are those with unigrams *e, o, i,* and *a* followed by space "_", thus, corresponding to words finished in vowel, as it occurs in Latin languages, and specially in Italian. In addition, is important to realize that bigrams starting with space "_" and followed by consonants *d, s* and *c* are also quite common, which corresponds to words started with these consonants.

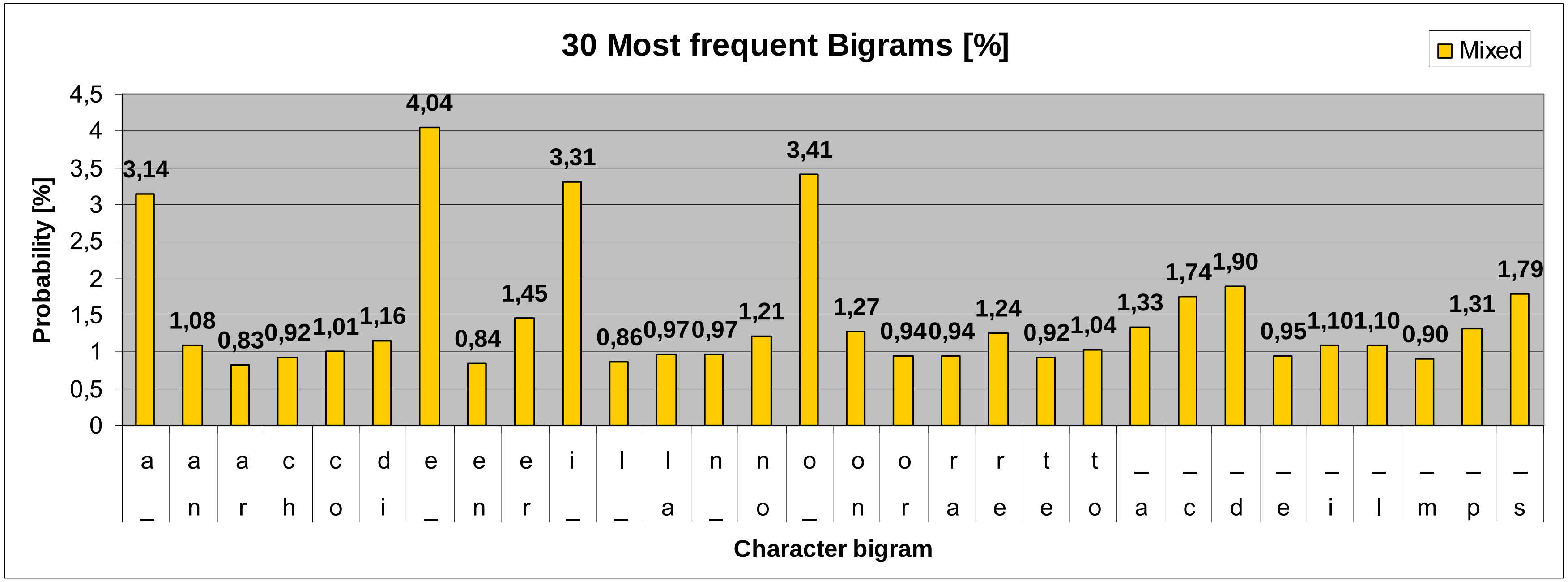


Fig. 3.3 Group of most frequent character bigrams in percentage above all bigrams in mixed training set

Secondly, **Fig. 3.4** shows also a distribution of the 30 most probable character bigrams for novels training set. Reductions of stored bigram information can be applied knowing that:

- A 30,72% of bigrams have 0 occurrences.
- A 53,36% of bigrams (including those with 0 occurrences) have 4 magnitude orders less than the most common bigrams.

If this is taken into consideration, those bigrams can be suppressed, remaining a 99.98% of bigram occurrences. The remaining percentage of bigrams, is mostly covered by only the 30 most probable bigrams, which are the 44,22% of all 27^2 bigrams.

As observed in most common bigrams of mixed training set, in novels training set can also be observed that most probable bigrams are those with unigrams *e, o, i,* and *a* followed by space "_". In addition, bigrams starting with space "_" and followed by consonants *d, s* and *c* are also quite common, as occurred in **Fig. 3.3**.

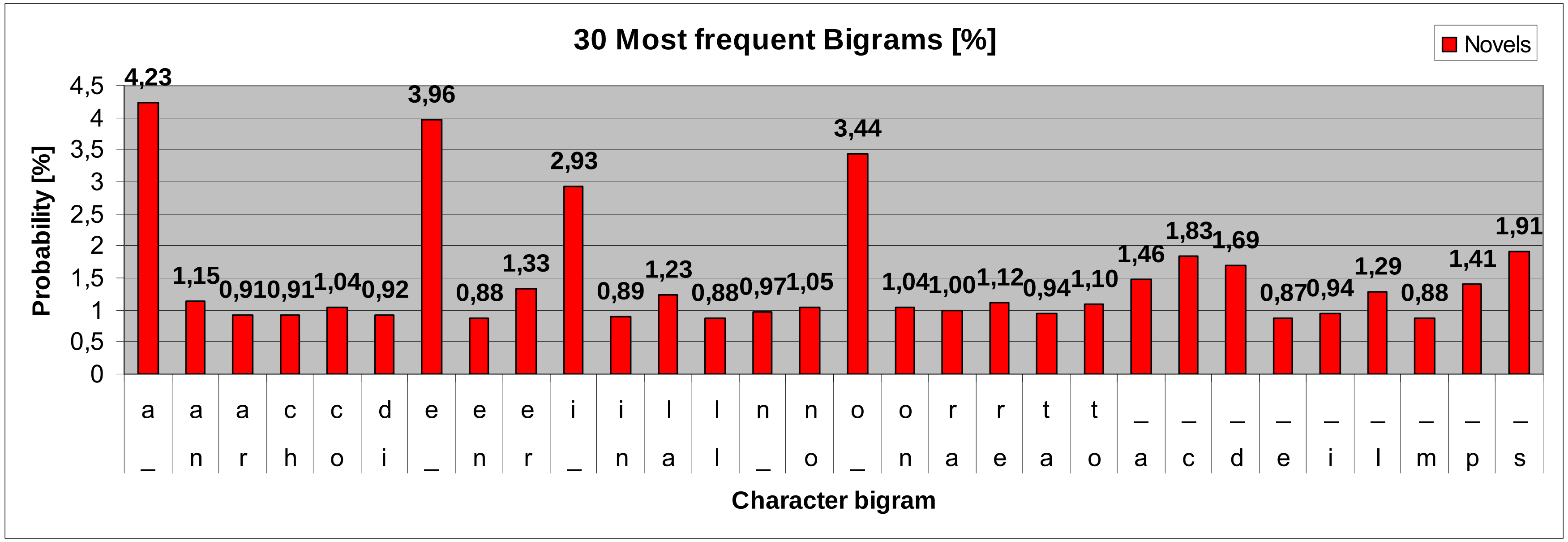


Fig. 3.4 Group of most frequent character bigrams in percentage above all bigrams in novels training set

3.2.2.3. Character trigrams

Fig. 3.5 shows a distribution of the 30 most probable character trigrams for mixed training set. Reductions of stored trigram information can be applied knowing that:

- A 67,37% of trigrams have 0 occurrences.
- A 79,59% of trigrams (including those with 0 occurrences) have 4 magnitude orders less than the most common bigrams.

If this is taken into consideration, those trigrams can be suppressed, remaining a 99.93% of trigram occurrences. The remaining percentage of trigrams, is mostly covered by only the 30 most probable trigrams, which are the 15,21% of all 27^3 trigrams. It can be observed that in those 30 most common trigrams, there is less weight of total trigram percentage in reference to it was in bigrams (15,21% in trigrams in front of 43,66% in bigrams). This decrease is due to the fact that in character trigrams information is concentrated in a smaller group, thus, there is not so much variety.

It can be observed that most probable trigrams are those corresponding to: words started with *di* or preposition *di* (trigram "*_ d i*"); words finished in *to* (trigram "*t o _*") as many of Italian words actually does, concretely, verbal forms; words finished in *re* (trigram "*r e _*") as many of Italian verbs does; and words finished in *la* (trigram "*l a _*").

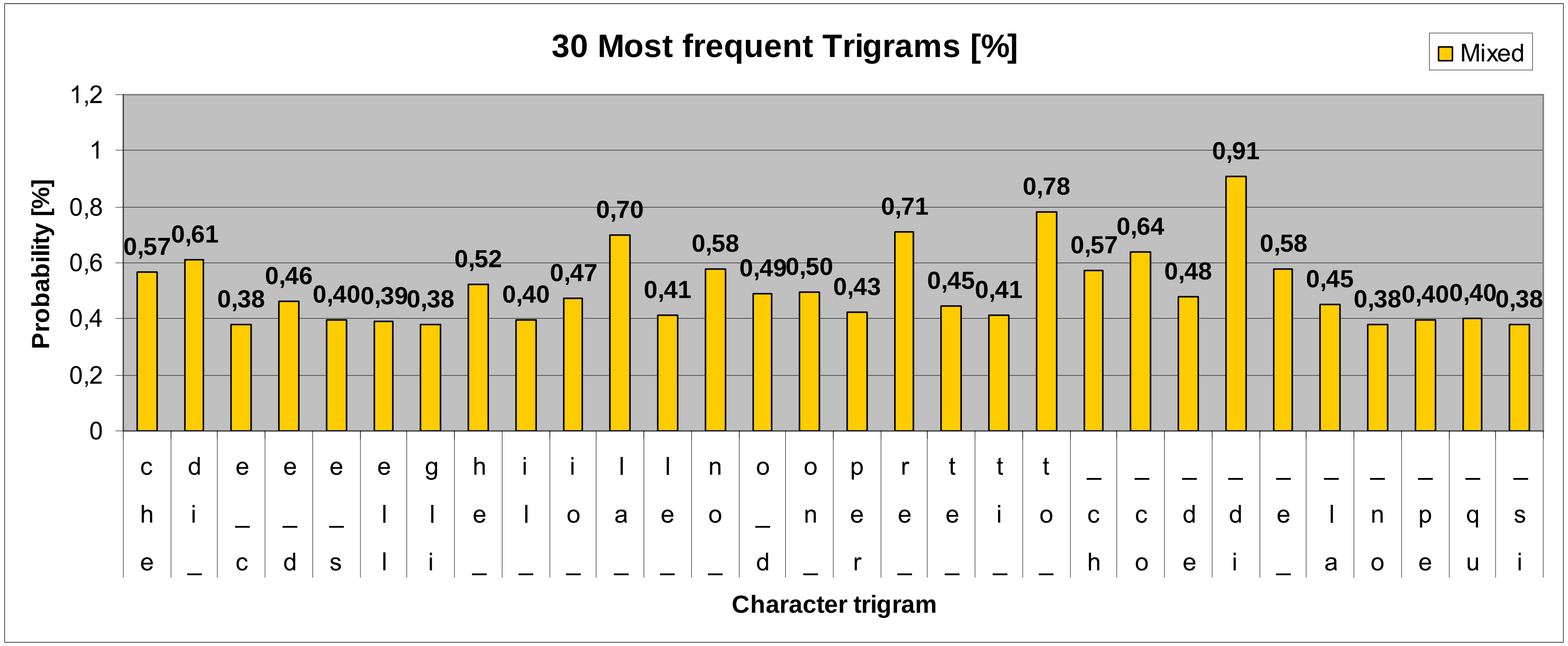


Fig. 3.5 Group of most frequent character trigrams in percentage above all trigrams in mixed training set

Fig. 3.6 also shows a distribution of the 30 most probable character trigrams for novels training set. Reductions of stored trigram information can be applied knowing that:

- A 76,49% of trigrams have 0 occurrences.
- A 82,17% of trigrams (including those with 0 occurrences) have 4 magnitude orders less than the most common bigrams.

If this is taken into consideration, those trigrams can be suppressed, remaining a 99.93% of trigram occurrences. The remaining percentage of trigrams, is mostly covered by only the 30 most probable trigrams, which are the 11,44% of all 27^3 trigrams. It can be observed that in those 30 most common trigrams, there is less weight of total trigram percentage in reference to it was in bigrams (11,44% in trigrams in front of 44,22% in bigrams). As it occurred in mixed training set, this decrease is due to the fact that in character trigrams information is concentrated in a smaller group, thus, there is not so much variety.

It can be observed that some of the most probable trigrams are different from those appeared in Fig. 3.5. In this case, some of the most common correspond to: words finished in *to* (trigram "*t o _*"); words finished in *la* (trigram "*l a _*"); words started with *di* or preposition *di* (trigram "*_ d i*"); and words started with *co* (trigram "*c o _*").

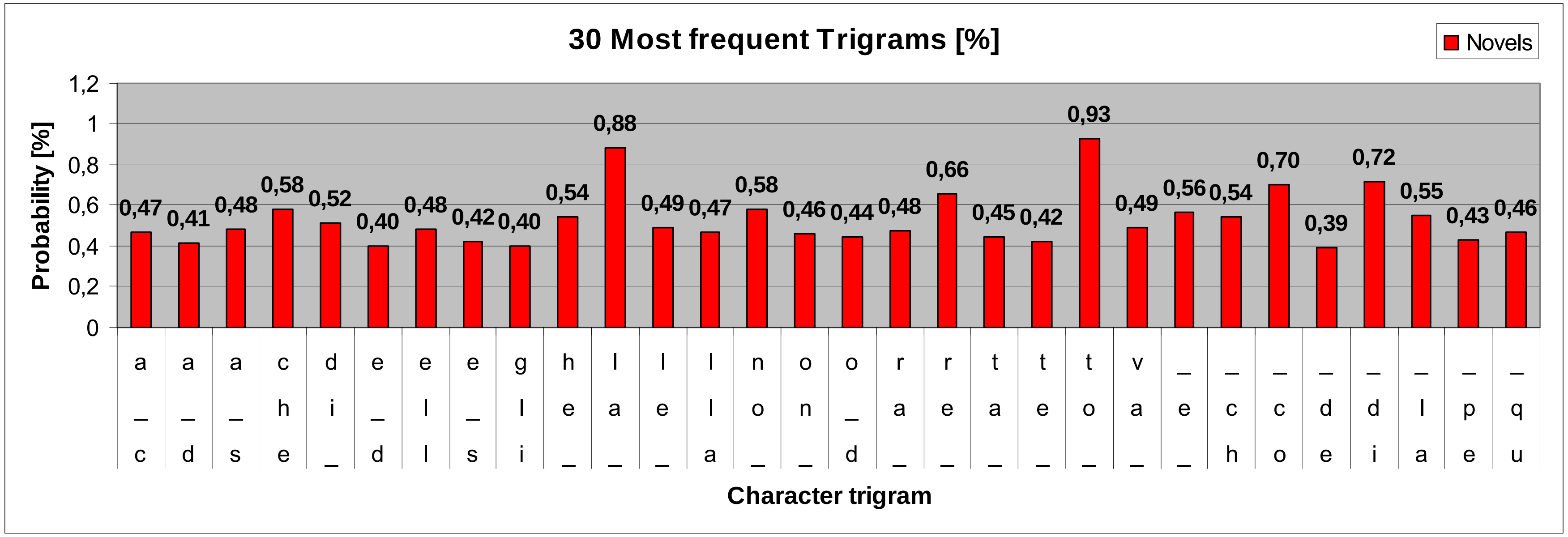
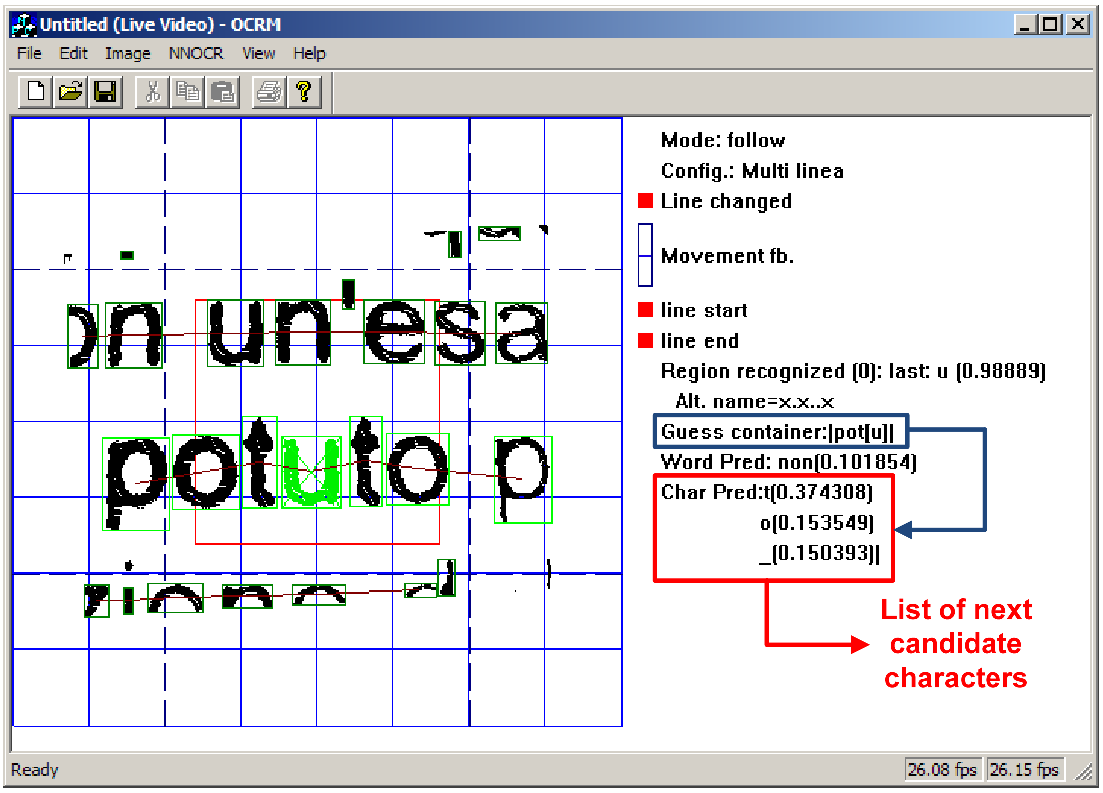


Fig. 3.6 Group of most frequent character trigrams in percentage above all trigrams in novels training set

3.3. Results

Firstly, it is important to notice that due to the fact that this is an experimental project, the prototype conditions of use have been considered ideal. Those are referring to the “scanner” movement over a text (finding line beginning or ending, line changing, etc).

Character predictions are added to the whole system application.
On the one hand, for a given group of two characters, language model is applied to offer a **list** of most probable next characters (a quick sort algorithm is applied to get the most probable characters). This is shown in **Fig. 3.7**, while characters are being recognized, the list is updated in real-time offering the most probable next character, with its probability. Concretely, for the bigram ***tu*** recognized, the list gives ***t*** as the next more probable character, also offering ***o*** and _.

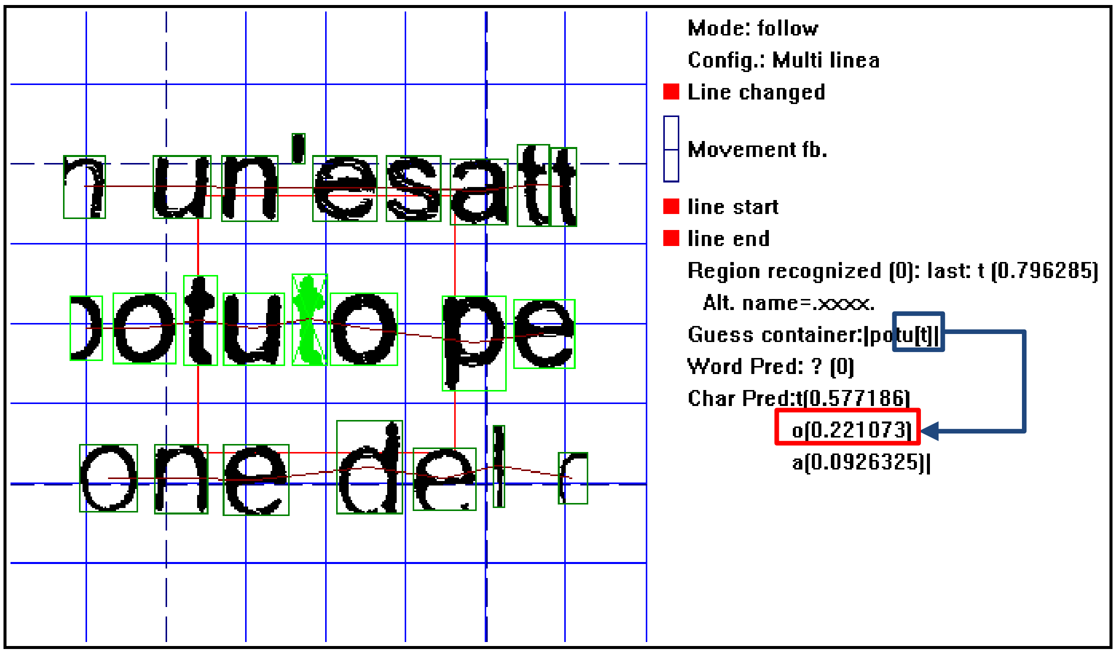

Untitled (Live Video) - OCRM
File Edit Image NNOCR View Help
List of next candidate characters

```
Mode: follow
Config.: Multi linea
Line changed
Movement fb.
line start
line end
Region recognized (0): last: u (0.98889)
  Alt. name=x.x..x
Guess container:|pot[u]|
Word Pred: non(0.101854)
Char Pred:t(0.374308)
          o(0.153549)
          _(0.150393)|
```

```
Ready                                   26.08 fps  26.15 fps
```

Fig. 3.7 Detail of character prediction in OCRM graphic interface

In **Fig. 3.8**, the list is updated when next character is recognized, having ***t***, the prediction system offers 3 new next possible characters. Concretely, the list gives ***t*** as the next more probable character, also offering ***o*** and ***a***.



```
   Mode: follow
   Config.: Multi linea
🟥 Line changed

   Movement fb.

🟥 line start
🟥 line end
   Region recognized (0): last: t (0.796285)
     Alt. name=.xxxx.
   Guess container:|potu[t]|
   Word Pred: ? (0)
   Char Pred:t(0.577186)
             o(0.221073)
             a(0.0926325)|
```

Fig. 3.8 Next character recognized in real-time
and detail of list of characters candidates predicted

On the other hand, when recognition system is unable to give a character recognized, prediction system gives the most probable character. This is useful to avoid misinterpretations when a text is read, due to the fact that it is a better solution to give some recognition although it is possibly not the correct one, instead of not giving any kind of recognition, which occurred in system developed in [2].

In order to evaluate the model, has been defined a group of characters as a word, which means each GoC corresponds to a group of characters separated by spaces (_).

Nevertheless, to evaluate the performance of the model applied to predict characters is a difficult task to achieve due to the fact that lots of GoC have to be tested for prediction. To have a valid evaluation of the real behavior of the model it is necessary to test the system with *corpora* not present in the training data set.

These additional *corpora* is normally called test *corpora*, which is a small part of the initial data not present in training data, which has been selected randomly. Therefore, it can be said that training set is a very good sample of the test data.

To obtain a valid evaluation, test data has to have a minimum length of 5% of length of training data, thus:

- Characters of test data for mixed training set would be approximately more than 435k characters.
- Characters of test data for novels training set would be approximately more than 320k characters.

Consequently, this is a very costly manual task due to the fact that a user has to move the prototype over a text of those amounts of characters to give a useful feedback.

In spite of the exposed requirements, an approximated evaluation has been realized in a text of 60 characters: *“moderare il passo ma soltanto badando di portare sempre tutti e due i piedi”*.

After a prediction is done based on a character recognized, a comparison between the 3 most probable predicted characters and the character recognized before is made. If any of those 3 is the same as character recognized before, it is considered that character prediction is well-done.

If none of those 3 predicted characters is the same as the one recognized before, the predicted one is assigned as the one of higher probability.

This information is passed to compute the evaluation measure, which are cross-entropy *H(T)* and perplexity *PP(T)*.

$$H(T) = -\frac{1}{C_T}\log_2\left(p(T)\right) \qquad \textbf{(3.3)}$$

$$PP_p(T) = 2^{H(T)} \qquad \textbf{(3.4)}$$

where,

C_T is the length of text *T* in terms of characters

$$p(T) = \prod_{i=1}^{L} p(GoC_i) = \prod_{i=1}^{L}\left[\prod_{i=1}^{k} p\left(c_i \middle| c_{i-2}\ c_{i-1}\right)\right]$$

where,

p(GoC) corresponds to the probability of a group of characters.

Those values depends directly of the recognition system results, thus, in every testing process, there is a variation that puts in doubt the interpretation of those values.

Table 3.2 Intuitive Cross-Entropy and Perplexity of character prediction

Training set	Cross-Entropy	Perplexity
Mixed	2,7603	6,7755
Novels	2,3599	5,1334

Chapter 4

Word-based Prediction

The aim of this chapter is to describe the methods used to predict words while character recognition and prediction processes are running.

As it has been exposed along this paper, the general goal of the whole project (see chapter 1) is to increase system robustness reaching a system able to allow blind people to read the most similar possible to as they read text in Braille. The fact of having statistical information of words, which are the basic unit within sentences, is the starting point of text context recognition and correction.

4.1. Used model

The statistical language model used to is the n-gram model with n = 2. As stated in chapter 2 and extrapolated in word terms, bigram **word** modeling is defined by equation (2.9), which is,

$$p(GoW)\big|_{BIGRAM} \approx \prod_{i=1}^{k} p_{MLE}\left(w_i \middle| w_{i-1}\right) \qquad \textbf{(4.1)}$$

where,

$$p\left(w_i \middle| w_{i-1}\right)_{MLE} = \frac{count\left(w_{i-1}\ w_i\right)}{count\left(w_{i-1}\right)} \qquad \textbf{(4.2)}$$

In this case, the probability of a word depends on the identity of the preceding word, ruled by the maximum likelihood estimate of $p_{MLE}\left(w_i \middle| w_{i-1}\right)$.

As exposed in section 2.2.2, in word-level prediction is necessary to add smoothing techniques to applied model. In this case, the bigram model is linearly interpolated with a unigram word model,

$$p_{\text{int}}\left(w_i \middle| w_{i-1}\right) = \lambda \cdot p_{MLE}\big|_{bigram}\left(w_i \middle| w_{i-1}\right) + (1-\lambda)\cdot p_{MLE}\big|_{unigram}\left(w_{i-1}\right) = \\ = \lambda \cdot \frac{count\left(w_{i-1}\ w_i\right)}{count\left(w_{i-1}\right)} + (1-\lambda)\cdot \frac{count\left(w_{i-1}\right)}{N_w} \tag{4.3}$$

where λ is fixed to 0,9, giving a weight of 90% to probability of word bigram and a 10% to probability of word unigram.

Thus, the model used to predict words is the **statistical word bigram model** with linear interpolation, expressed by the following equation,

$$\boxed{p\left(GoW\right)\big|_{BIGRAM} \approx \prod_{i=1}^{k} p_{\text{int}}\left(w_i \middle| w_{i-1}\right)} \tag{4.4}$$

getting the behavior that bigrams involving common words are assigned higher probabilities.

4.2. Model training

4.2.1. Data set

Table 4.1 show the constructed set of *corpora*, which have been also extracted from [14]. Its aim is to train the bigram word model. The training set is composed of text files (*.txt) of novels. The selection criterion of this type of texts is that book novels are a vast amount of existing readable information.

Table 4.1 Training set specifications

Text file	# Words	# Characters
L'assassinio di Via Belpoggio (Italo Svevo)	23827	146881
La Divina Commedia (Dante Alighieri)	101493	535554
Sort of restaurant menus	1961	12743
TOTAL	**127281**	695178

4.2.2. Model construction

As stated also in character prediction, training *corpora* is used to construct the model which assigns occurrences corresponding to each appeared word unigrams and bigrams. Thus, a structure of dynamical vectors has been built to store word occurrences as can be seen in **Fig. 4.1**.

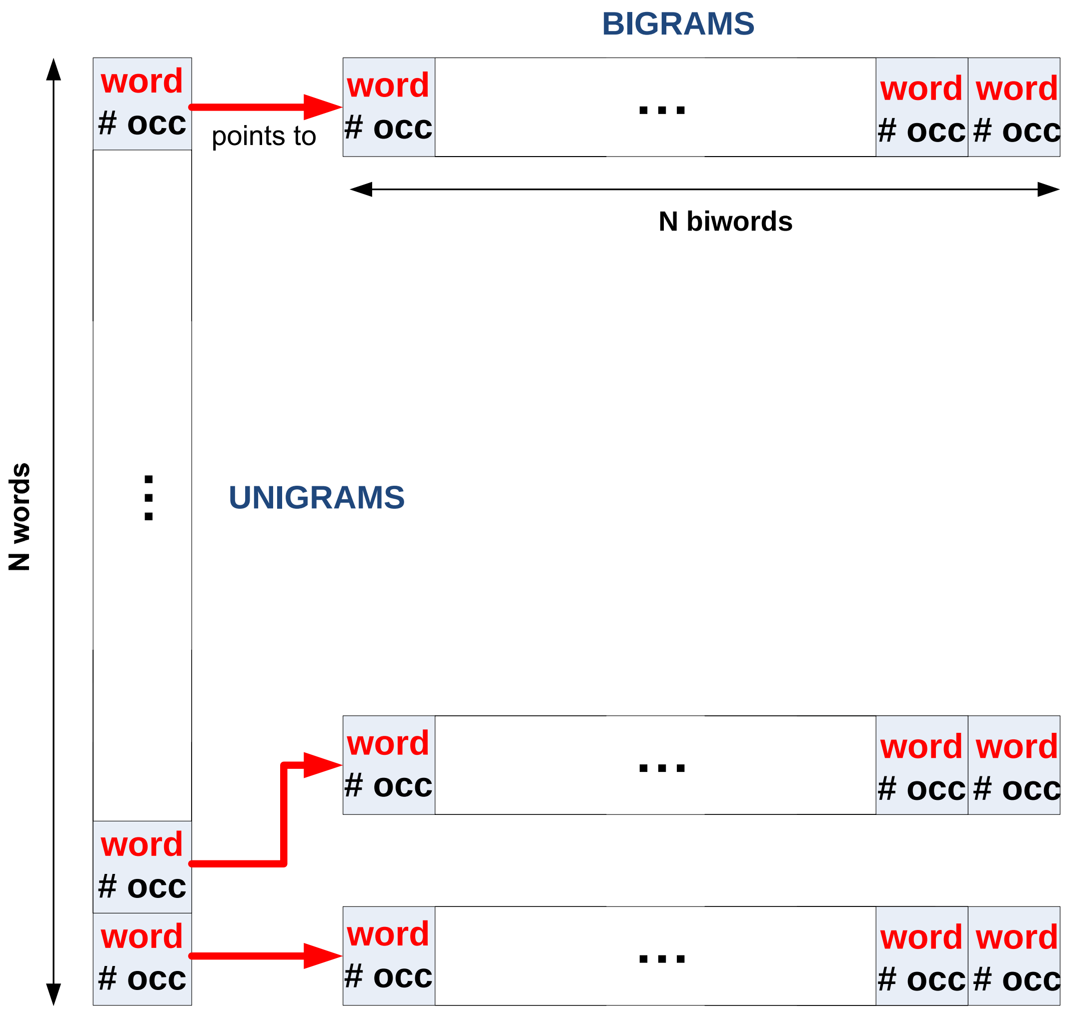


Fig. 4.1 Structure for storing word unigrams and bigrams

While words are appearing are being stored in the structure, simultaneously occurs with bigrams. When unigrams and bigrams are already appeared, its occurrences are incremented. Parameters N words and N biwords, are manually fixed.

The following **Fig. 4.2** show, as an example, text as the input of training system which assigns occurrences to each word unigram and bigram defined below.

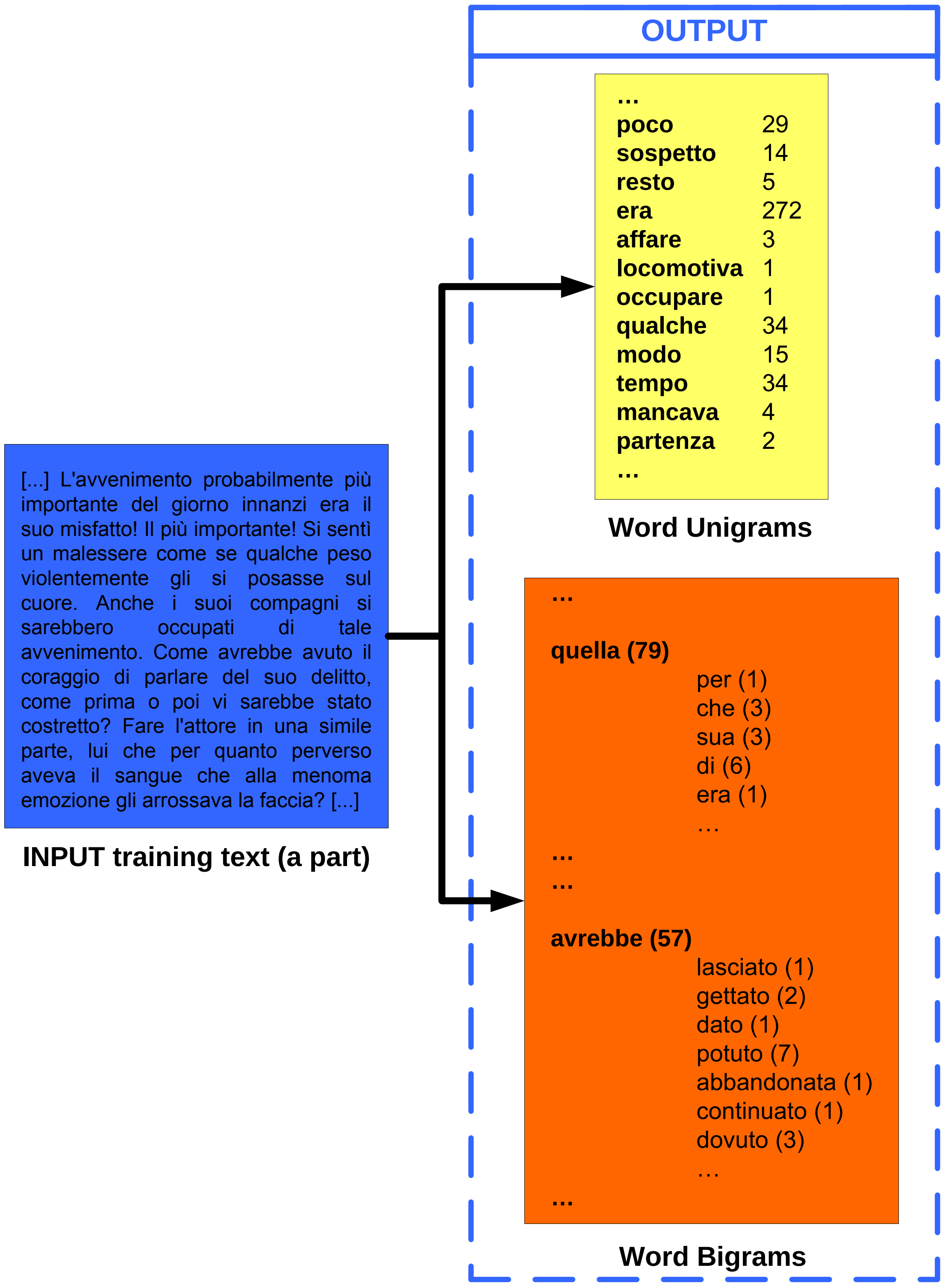
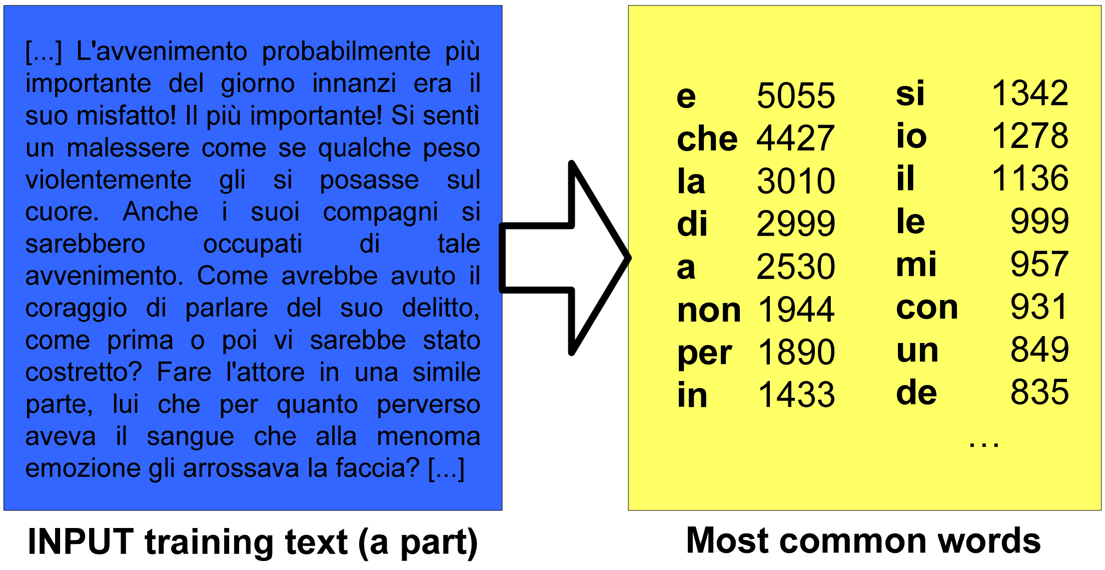

OUTPUT
...
poco 29
sospetto 14
resto 5
era 272
affare 3
locomotiva 1
occupare 1
qualche 34
modo 15
tempo 34
mancava 4
partenza 2
...
Word Unigrams
...
quella (79)
per (1)
che (3)
sua (3)
di (6)
era (1)
…
...
...
avrebbe (57)
lasciato (1)
gettato (2)
dato (1)
potuto (7)
abbandonata (1)
continuato (1)
dovuto (3)
…
...
Word Bigrams
[...] L'avvenimento probabilmente più importante del giorno innanzi era il suo misfatto! Il più importante! Si sentì un malessere come se qualche peso violentemente gli si posasse sul cuore. Anche i suoi compagni si sarebbero occupati di tale avvenimento. Come avrebbe avuto il coraggio di parlare del suo delitto, come prima o poi vi sarebbe stato costretto? Fare l'attore in una simile parte, lui che per quanto perverso aveva il sangue che alla menoma emozione gli arrossava la faccia? [...]
INPUT training text (a part)

Fig. 4.2 Constructing a word n-gram list

As can be observed in **Figs. 4.1** and **4.2**, each word unigram has an associated list of words, both forming a word bigram.

As in character model training, training step of word model also runs when software application is initialized. Therefore, while characters are being recognized, the access to statistics of words is immediate. **Fig. 4.3** shows the most common words in training set specified in **Table 4.1**.



[...] L'avvenimento probabilmente più importante del giorno innanzi era il suo misfatto! Il più importante! Si sentì un malessere come se qualche peso violentemente gli si posasse sul cuore. Anche i suoi compagni si sarebbero occupati di tale avvenimento. Come avrebbe avuto il coraggio di parlare del suo delitto, come prima o poi vi sarebbe stato costretto? Fare l'attore in una simile parte, lui che per quanto perverso aveva il sangue che alla menoma emozione gli arrossava la faccia? [...]

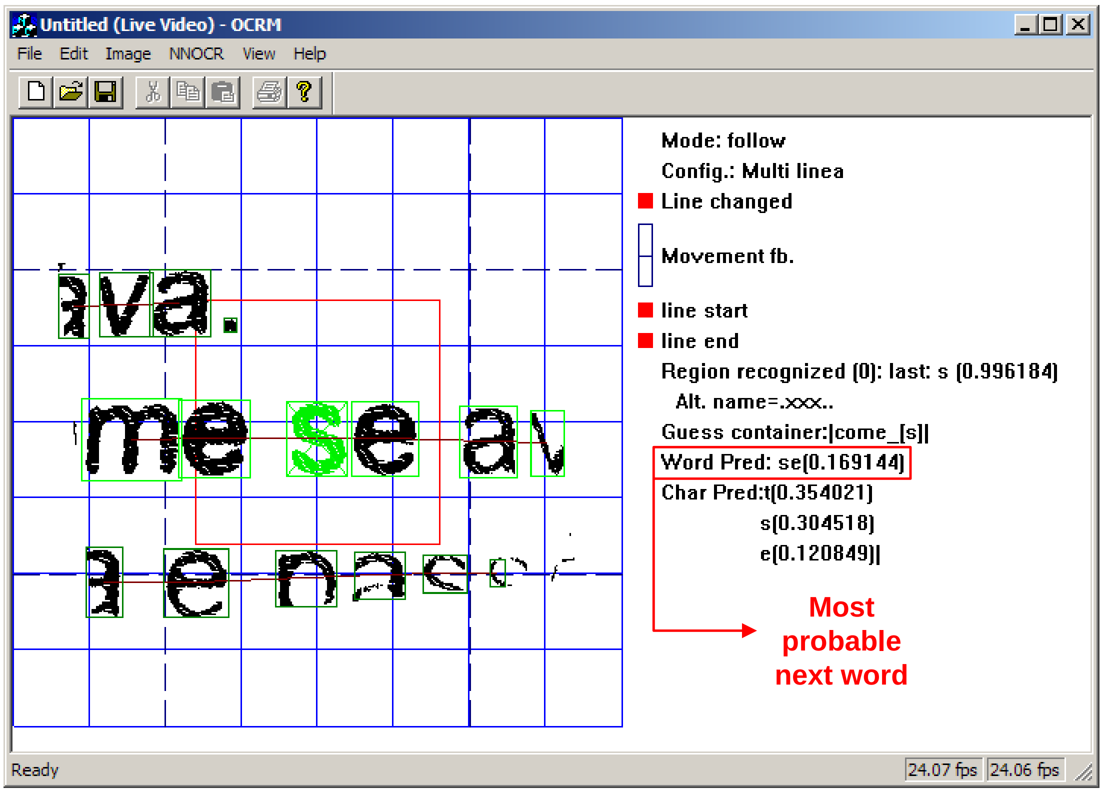
e	5055	**si**	1342
che	4427	**io**	1278
la	3010	**il**	1136
di	2999	**le**	999
a	2530	**mi**	957
non	1944	**con**	931
per	1890	**un**	849
in	1433	**de**	835
			…

Fig. 4.3 Words ordered by number of occurrences

4.3. Results

As stated on chapter 3, the prototype conditions of use have been considered ideal. Those are referring to the “scanner” movement over a text (finding line beginning or ending, line changing, etc).

Word predictions are added to the whole system application.
For a given word recognized, most probable next word is given. This is shown in **Fig. 4.4**, while a group of characters that form a word (space between them) are being recognized, the predicted word is updated, giving also its probability of appearance. Concretely, for the group of characters recognized forming the word ***come***, the model gives ***se*** as the next more probable word.



Mode: follow
Config.: Multi linea
🟥 Line changed
Movement fb.
🟥 line start
🟥 line end
Region recognized (0): last: s (0.996184)
Alt. name=.xxx..
Guess container:|come_[s]|
Word Pred: se(0.169144)
Char Pred:t(0.354021)
s(0.304518)
e(0.120849)|

Fig. 4.4 Detail of word prediction in OCRM graphic interface

In order to evaluate the model, it is necessary to define a group of words as a sentence, which means each GoW corresponds to a group of words in the same line in text.

As it has been explained in character prediction system, to evaluate the performance of the model applied to predict words is a difficult task to achieve due to the fact that a lots of GoW have to be tested for prediction. To have a valid evaluation of the real behavior of the model it is necessary to test the system with text *corpus* not present in the training data set, called test *corpus*.

To obtain a valid evaluation, test data has to have a minimum length of 5% of length of training data, thus, for that constructed in this model, would be approximately more than 850 words. In terms of characters means an average amount of more than 34k characters.

Consequently, this is a very costly manual task due to the fact that a user has to move the prototype over a text of those amounts of words (concretely, character by character) to give a useful feedback.

Although testing is not realized through the prediction system, procedures of evaluation are specified. Which are the computation of the evaluation measures, cross-entropy *H(T)* and perplexity *PP(T)*.

$$H(T) = -\frac{1}{W_T} \log_2 \left(p(T) \right)$$

(4.5)

$$PP_p(T) = 2^{H(T)}$$

(4.6)

where, particularly,

W_T is the length of text *T* in terms of words.

$$p(T) = \prod_{i=1}^{L} p(GoW_i) = \prod_{i=1}^{L} \left[\prod_{i=1}^{k} p\left(w_i \middle| \mathrm{w}_{i-1} \right) \right]$$

where,

p(GoW) corresponds to the probability of a group of words.

Chapter 5
Text Correction

This chapter is devoted to explain how statistical language models, based on prediction, can be applied to correct recognition errors made by the present system. In addition, other correction techniques, which can also be applied, are stated.

5.1. Statistical language models used as correction methods

In [15] an extensive study of text correction techniques has been developed. Correction problem is considered in different points of view:

- Non-word error detection
- Isolated-word error correction
- Context dependent word correction

5.1.1. Within-word context error detection

Due to the fact that recognition system is based on OCR, mostly errors occurs when confusing characters of similar features, such as *f* and *t*, *i* and *l*.

To detect such errors within words, the constructed **character trigrams** in the present paper are useful. The reason is that such errors tend to occur in improbable trigrams.

As it has been studied in chapter 3, statistical features of training (taking as reference novels training set) showed that a 31% of all possible bigrams never occur and a 76% of all possible trigrams neither occur. In addition, if

one letter in a word is substituted randomly by another of the 25 letters in alphabet, will imply in 70% of cases at least one new bigram with 0 probability [15].
The above facts can be utilized for detecting such errors.

In addition, character trigrams capture the lexical features of a language. Therefore, this information can be applied to correct characters. For example in Italian, the consonant *q* is always followed by the vowel *u*. If the current system recognizes a character different from *u*, the error can be detected and corrected.

Another method to detect such errors is based on dictionary lookup techniques. The aim of such techniques is to search in built dictionaries to detect possible errors. The access to dictionaries has to be efficient.

To investigate in this line, a different set of dictionaries has been constructed, with the particularity that includes verbs in different times and genres, not only in the infinitive form. A dictionary of almost 300k words and another one of 600k words are available, both separated in 26 text files containing words started with each alphabet letter. These created dictionaries will also be useful for further techniques.

5.1.2. Isolated word error correction

Character n-grams (trigrams, bigrams and unigrams) are also useful in word detection and correction methods. N-grams can capture the lexical syntax of a dictionary and suggest corrections. They can be used as:

- the access key into a dictionary to find possible candidate corrections
- lexical features for computing similarity measures

Therefore, additional character n-gram training is available in this paper focused on training a character trigram model with built dictionaries (see the above section). It can be observed in **Fig. 5.1**.

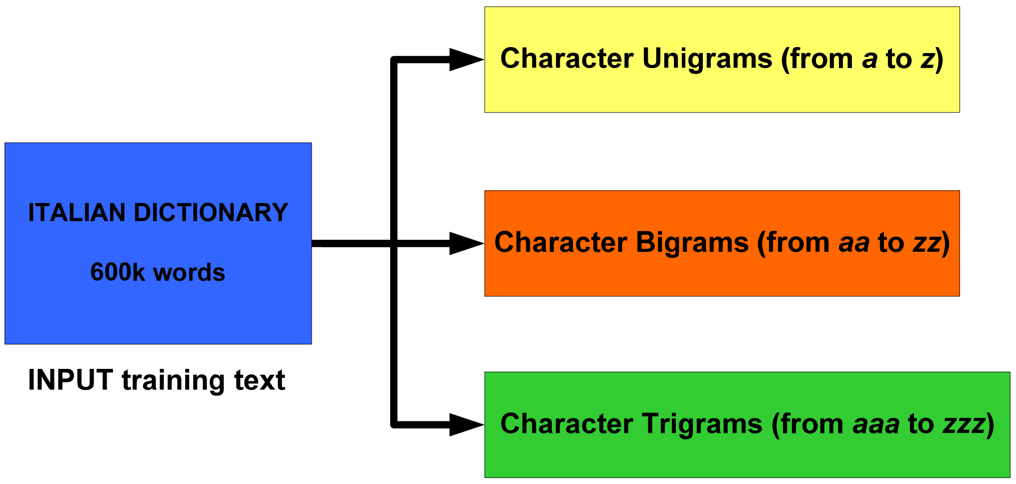

ITALIAN DICTIONARY
600k words
INPUT training text
Character Unigrams (from *a* to *z*)
Character Bigrams (from *aa* to *zz*)
Character Trigrams (from *aaa* to *zzz*)

Fig. 5.1 Constructing a character n-gram list

5.1.3. Context dependent word correction

Implemented word bigram model is useful to detect and correct context dependent words, due to the fact that the model obtains context information of text. The study realized in [16], can be useful as the starting point of the application of the work done in this paper to improve global system accuracy in terms of correction based on context.

Generally, all correction techniques use the *Levenshtein distance* as the base to correct elements in text. The *Levenshtein distance* between two words *V* and *W* is the minimal number of edit operations (substitutions, deletions or insertions) that are needed to **transform** *V* into *W*.

Chapter 6

Conclusions and future work

Allowing blind people to read printed information without the need of using special books in Braille is a challenging task. Building a system that allows them to read the most similar possible to as they read text printed in Braille is also a challenging work.

Firstly, in order to improve the reliability of the system developed in [2], a character and word prediction systems have been implemented and added to the existing system. Such systems are based on statistical language modeling.

On the one hand, the character prediction system is based on a trigram language model which it has been properly trained to give in real-time the next character predicted based on the last two preceding characters. This prediction system interacts in real-time with the recognition system.

On the other hand, the word prediction system is based on a bigram language interpolated model which gives in real-time the most probable following word based on the last word (which are the last recognized characters). The assignation of word predicted is based on language knowledge of the model.

These systems also establish the starting point for correction techniques.

In addition, a set of statistical language tools (SRILM package) have been prepared for possible future uses, such as improving built models. Furthermore, different digital Italian dictionaries have been built and separated into files for uses in correction techniques based on systems developed.

Finally, on the one hand, future lines of investigation would be those focused on improving implemented models.

For example, in word prediction, improvements should be done in terms of the implementation of methods to separate words into clusters, applying topic language models [17] to obtain more accurate prediction, etc. Allowing the model to identify specific lexicon and, still more a difficult task, to identify grammatical expressions.

On the other hand, future work should be focused on applying correction techniques to increase system reliability.

References

[1] World Health Organization blindness estimation.
Available at: http://www.who.int/en/

[2] Motto Ros, P., “Lettore ottico per non vedenti”, *Tesi di Laurea, Dipartimento di Elettronica, Politecnico di Torino*, 2005.

[3] *STIPER 2* - Istituto Nazionale di Fisica Nucleare.
Available at: http://www.to.infn.it/it/stiper-2

[4] *STIPER 2* - Istituto Superiore di Sanità. Available at:
http://www.iss.it/tesa/acco/cont.php?id=184&lang=1&tipo=18

[5] *STIPER 2* - Laboratorio di Neuronica. Dipartimento di Elettronica. Politecnico di Torino. Available at:
http://www.neuronica.polito.it/default.asp?view=research_image_stiper&lang=en

[6] Bellegarda J.R., “Statistical language model adaptation: review and perspectives”, *Speech communication,* vol. 42, No. 1, pp. 93-108, 2004.

[7] Lau, R. “Adaptive Statistical Language Modelling”, S.M. Thesis, *Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology*, 1994.

[8] Katz, S.M., “Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer”, *IEEE Transactions on Acoustics Speech and Signal Processing,* Vol. 35, No. 3, pp. 400-401, 1987.

[9] Kneser, R. and Ney, H., “Improved Backing-Off for M-gram Language Modeling”, *Proceedings IEEE ICASSP,* 1995.

[10] SRI Language Modeling toolkit.
Available at: http://www.speech.sri.com/projects/srilm/

[11] CMU-Cambridge Statistical Language Modeling toolkit.
Available at: http://www.speech.cs.cmu.edu/SLM_info.html

[12] Cygwin. Available at: http://www.cygwin.com/

[13] Srihari R. et alt. “Use of Language Models in Handwriting Recognition”, *Center of Excellence for Document Analysis and Recognition, The State University of New York*, 2007.

[14] Digital library. Available at: http://www.liberliber.it/biblioteca/index.htm

[15] Kukich K., “Techniques for Automatically Correcting Words in Text”, *ACM Computing Surveys*, Vol. 24, No. 4, pp. 377-439, 1992.

[16] Church, K. W. and Gale, W. A., “Probability scoring for spelling correction”, *Stat. Comput.*, No. 1, pp 93–103., 1991.

[17] Bhardwaj, A. et alt., “Topic based language models for OCR correction”, *ACM International Conference Proceeding Series; Proceedings of the second workshop on analytics for noisy unstructured text data*, Vol. 303, pp. 107-112, 2008.