UNIVERSIDAD POLITÉCNICA DE VALENCIA
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN


# Interactive Pattern Recognition applied to Natural Language Processing

Thesis
presented by Luis Rodríguez Ruiz
supervised by Dr. Ismael García Varea and Dr. Enrique Vidal Ruiz

June 15, 2010

# Interactive Pattern Recognition applied to Natural Language Processing

Luis Rodríguez Ruiz

Thesis performed under the supervision of doctors
Ismael García Varea and Enrique Vidal Ruiz
and presented at the Universidad Politécnica de Valencia
in partial fulfilment of the
requirements for the degree
Doctor en Informática

Valencia, June 15, 2010

# ACKNOWLEDGMENTS

Quiero aprovechar estas líneas para mostrar mi agradecimiento a todas las personas que de una forma o de otra han contribuido a que esta tesis se haya hecho realidad.

En primer lugar quiero agradecer sinceramente a mis directores de tesis todo su apoyo, paciencia y dedicación. Ismael, a quién hace tiempo que no considero simplemente mi director de tesis, sino un amigo y del que he aprendido muchas cosas, sobre todo en cuanto a traducción automática se refiere. En cuanto a Enrique, no puedo sino considerarlo como un auténtico maestro para mí. Gracias por enseñarme que el detalle es realmente importante y por reconducir mis raras e improvisadas propuestas. En definitiva, si de alguna forma ahora puedo considerarme un investigador, es fundamentalmente gracias a ellos. También me gustaría incluir aquí a Francisco Casacuberta, con quien he compartido varios proyectos y artículos y del que también he tenido oportunidad de aprender muchas cosas.

No quiero dejar pasar la oportunidad de agradecer a todos los que de uno u otro modo han compartido conmigo su aprendizaje en el mundo de la investigación. Empezando con mis compañeros en el proyecto TT2: Antonio, Elsa y Jorge, con quien tantos momentos entrañables compartí. Extiendo este agradecimiento a todos los miembros del ITI que me brindaron una extraordinaria acogida durante mis casi tres años en Valencia: Alejandro (quien siempre tiene algún truco nuevo que enseñarte), José Ramón, Dani (con quien siempre se mantienen conversaciones productivas), Javi, Jose, Jorge, Vicent y a todos los miembros del grupo PRHLT. También quiero agradecer a todos los integrantes del grupo SIMD, especialmente a sus directores José Antonio y José Miguel por acogerme a mi regreso de Valencia. Por último, quiero también mostrar mi agradecimiento a José Oncina, por encontrar un algoritmo que hace mejores los resultados de predicción de texto aquí presentados.

En el aspecto personal, quiero agradecer a mis padres todo su apoyo y cariño en las diferentes etapas de mi vida. También al resto de mi familia. A todos mis amigos, por saber perdonar mi falta de dedicación con ellos debido a lo absorbente de este trabajo. Por último, mi más profundo agradecimiento a Sonia, por todo su apoyo y amor durante estos años tan complicados.

Finalmente, y citando a los geniales “Monty Python”: *and now, for something completely different ...*

Luis Rodríguez Ruiz
Albacete, June 15, 2010

# ABSTRACT

This thesis is about Pattern Recognition. In the last decades, huge efforts have been made to develop automatic systems able to rival human capabilities in this field. Although these systems achieve high productivity rates, they are not precise enough in most situations. Humans, on the contrary, are very accurate but comparatively quite slower. This poses an interesting question: the possibility of benefiting from both worlds by constructing cooperative systems.

This thesis presents diverse contributions to this kind of collaborative approach. The point is to improve the Pattern Recognition systems by properly introducing a human operator into the system. We call this *Interactive Pattern Recognition* (IPR).

Firstly, a general proposal for IPR will be stated. The aim is to develop a framework to easily derive new applications in this area. Some interesting IPR issues are also introduced. Multi-modality or adaptive learning are examples of extensions that can naturally fit into IPR.

In the second place, we will focus on a specific application. A novel method to obtain high quality speech transcriptions (CAST, *Computer Assisted Speech Transcription*). We will start by proposing a CAST formalization and, next, we will cope with different implementation alternatives. Practical issues, as the system response time, will be also taken into account, in order to allow for a practical implementation of CAST. Word graphs and probabilistic error correcting parsing are tools that will be used to reach an alternative formulation that allows for the use of CAST in a real scenario.

Afterwards, a *special* application within the general IPR framework will be discussed. This is intended to test the IPR capabilities in an *extreme* environment, where no input pattern is available and the system only has access to the user actions to produce a hypothesis. Specifically, we will focus here on providing assistance in the problem of text generation. The use of adaptive learning in this scenario will be emphasized. Besides, two derived applications will be also considered. Notably, the use of text prediction for information retrieval systems.

In addition, we will pose an interesting question about IPR systems. The inclusion of multi-modality as a natural part of IPR. The design of a speech input interface for Computer Assisted Translation (CAT) will be addressed. To this end, we will describe several interaction scenarios, which facilitate the speech recognition process by taking advantage of the CAT environment.

Finally, a set of prototypes that include the main features of the work here developed will be presented. The main motivation is to provide real examples about the feasibility of implementing the techniques here described.

# RESUMEN

El presente trabajo versa sobre Reconocimiento de Formas. En las últimas décadas, se han destinado numerosos esfuerzos en construir sistemas automáticos capaces de competir con las habilidades humanas en este campo. Aunque dichos sistemas son capaces de obtener niveles de productividad muy altos no son lo suficientemente precisos en muchos casos. Los seres humanos, por otra parte, resuelven este problema de forma bastante precisa, aunque no pueden competir en cuanto a velocidad. Este hecho plantea un problema interesante: la posibilidad de combinar ambas aproximaciones construyendo sistemas cooperativos.

Esta tesis se centra en presentar diferentes contribuciones a una nueva propuesta encuadrada dentro de este tipo de sistemas colaborativos. Para ello, se propone incluir al usuario como parte del propio sistema. Esta aproximación se conoce con el nombre de Reconocimiento Asistido de Formas (IPR, *Interactive Pattern Recognition*).

En primer lugar, se propondrá una formulación general para el problema del Reconocimiento Asistido de Formas. Se pretende, de esta manera, desarrollar un marco formal que permita el desarrollo de nuevas aplicaciones dentro de este campo. Por otra parte, se discutirán ciertos aspectos generales, relevantes dentro del marco de IPR. Cuestiones como la multi-modalidad o el aprendizaje adaptativo constituyen extensiones naturales al problema en cuestión.

En segundo lugar, se desarrollará una nueva aplicación destinada a obtener transcripciones del habla de calidad. Para ello, primeramente se estudiará una formalización de dicha aplicación para, más adelante, proponer diferentes alternativas de implementación. Se discutirán, además, diversos aspectos prácticos, como por ejemplo el tiempo de respuesta que presenta un sistema de este tipo. El uso de grafos de palabras y las técnicas de análisis sintáctico corrector de errores serán incluidas en una formulación alternativa encaminada a mejorar dicho tiempo de respuesta.

A continuación, se describirá un caso especial de aplicación, en la cual no se dispone de un patrón de entrada a reconocer y el sistema sólo puede basarse en las acciones realizadas por el usuario para generar nuevas hipótesis. Desde un punto de vista práctico, este enfoque pretende facilitar la generación de texto en diferentes situaciones. Además, se describirán dos aplicaciones derivadas de esta propuesta, destacando el uso de sistemas de generación de texto en sistemas de recuperación de información, que se presenta como una aproximación completamente nueva en este campo.

Por otra parte, se discutirá la inclusión de interfaces multi-modales en un sistema IPR. En concreto, se abordará el diseño de un interfaz basado en reconocimiento del habla para un sistema de traducción asistida. Se estudiarán, para ello, diferentes escenarios de interacción.

Por último, se presentará una serie de prototipos que implementan algunas de las técnicas aquí desarrolladas, con el objeto de mostrar su viabilidad como aplicaciones finales para el usuario.

# RESUM

El present treball versa sobre Reconeixement de formes. En les últimes dècades, s'han destinat nombrosos esforços a construir sistemes automàtics capaços de competir amb les habilitats humanes en aquest camp. Encara que aquests sistemes són capaços d'obtenir nivells de productivitat molt alts, no són prou precisos en molts casos. Els éssers humans, per altra banda, resolen aquest problema de forma prou precisa, encara que no poden competir quant a velocitat. Aquest fet planteja un problema interessant: la possibilitat de combinar ambdues aproximacions construint sistemes cooperatius.

Aquesta tesi se centra a presentar diferents contribucions a una nova proposta enquadrada dins d'aquest tipus de sistemes col·laboratius. Amb aquesta finalitat, es proposa incloure a l'usuari com part del propi sistema. Aquesta aproximació es coneix amb el nom de Reconeixement Assistit de Formes (IPR, *Interactive Pattern Recognition*).

En primer lloc, es proposarà una formulació general per al problema del Reconeixement Assistit de Formes. Es pretén, d'aquesta manera, desenvolupar un marc formal que permeta el desenvolupament de noves aplicacions dins d'aquest camp. D'altra banda, es discutiran certs aspectes generals, rellevants dins del marc de IPR. Qüestions com la multi-modalitat o l'aprenentatge adaptatiu constitueixen extensions naturals al problema en qüestió.

En segon lloc, es desenvoluparà una nova aplicació destinada a obtenir transcripcions del parla de qualitat. Amb aquesta finalitat, en primer lloc s'estudiarà una formalització d'aquesta aplicació per a, més endavant, proposar diferents alternatives d'implementació. Es discutiran, a més a més, diversos aspectes pràctics, com ara el temps de resposta que presenta un sistema d'aquest tipus. L'ús de grafs de paraules i les tècniques d'anàlisi sintàctica correctores d'errors seran incloses en una formulació alternativa encaminada a millorar aquest temps de resposta.

A continuació, es descriurà un cas especial d'aplicació, en la qual no es disposa d'un patró d'entrada a reconèixer i el sistema només pot basar-se en les accions realitzades per l'usuari per a generar noves hipòtesis. Des d'un punt de vista pràctic, aquest enfocament pretén facilitar la generació de text en diferents situacions. A més a més, es descriuran dues aplicacions derivades d'aquesta proposta, destacant l'ús de sistemes de generació de text en sistemes de recuperació d'informació, que es presenta com una aproximació completament nova en aquest camp.

Per altra banda, es discutirà la inclusió d'interfícies multi-modals en un sistema IPR. En concret, s'abordarà el disseny d'una interfície basada en reconeixement de la parla per a un sistema de traducció assistida. S'estudiaran, amb aquesta finalitat, diferents escenaris d'interacció .

Finalment, es presentarà una sèrie de prototips que implementen algunes de les tècniques ací desenvolupades, amb l'objecte de mostrar la seua viabilitat com aplicacions finals per a l'usuari.

# PREFACE

Pattern Recognition is a very natural task for us, human beings. Million years ago, accurately recognizing faces could represent the difference between life and death. Confusing an enemy or predator for a family member constituted a terrible and usually deadly mistake. Evolutively, we have acquired mechanisms able to extract and identify different kind of patterns from our environment. Speech and language gave us a considerable advantage over our competitors, since it allowed us to communicate in a very precise and efficient way. Color and shape identification provided us with a way to distinguish between healthy and poisonous food. These skills were fundamental for our survival and, because of that, we are highly-accurate pattern decoders.

Since sufficient computation power was available, we have tried to emulate these capabilities by constructing artificial systems. Some success have been achieved so far but not to the extent to which we can replace a human operator in most cases. Instead, only in some specific and constrained situations we can fully rely on automatic Pattern Recognition.

This problem can prevent these system from being used in real tasks unless some human supervision is added to the process. An operator can be employed to correct the mistakes produced by the system. This can make sense when the results are precise enough, since these users only have to modify a small portion of the outcomes and these changes are easy to make. However, when the number of errors is too high and/or they are difficult to be identified and corrected, a utterly manual process can instead be more productive.

In some situations some mistakes can be assumed. This is generally the case of applications where the result itself is not what really matters but only some information that can be derived from it (for instance, when an automatic translator is used to get the gist of a text written in a foreign language). On the contrary, going back to the translation example, formal documents are not expected to have errors and perfect results are required in this case. We can cite the specific example of legal documents, where the consequences of an inaccurate translation can be really dramatic and, therefore, considering complete automation here is not possible at all.

Focusing, however, on fully automatic systems is not the only alternative. Instead, devising tools that can complement the work of a human operator (or "user") is quite worth it. The goal here is to keep the human Pattern Recognition skills while achieving a high productivity. Building semi-automatic, cooperative applications, where the user is an integral part of the system, is a way to accomplish this objective.

In this thesis, we are going to study some of the possibilities that this kind of paradigm can offer. Since Pattern Recognition is such a wide field, we are going to focus on the problem of Natural Language Processing. Speech recognition, assis-

tance in the generation of text documents or user interfaces based on natural language are interesting problems that will be addressed in this work. Clearly, these are only a few examples and a great number of new possibilities are ready to be explored nowadays.

The organization of this thesis is as follows:

**Chapter 1** presents an introduction to Pattern Recognition and Natural Language Processing. In addition, the main goals of the work are discussed and the specific tasks used in the experiments are described.

**Chapter 2** introduces the concept of *Interactive Pattern Recognition (IPR)*, describing in detail the fundamentals and the motivation of this proposal. Besides, a formal framework to develop IPR systems is proposed. Finally, different system architectures as well as important issues derived from this general framework are discussed.

**Chapter 3** is devoted to presenting a specific application of the IPR paradigm. The case of Computer Assisted Speech Transcription (CAST) is addressed. We include an extension from the well known statistical speech recognition problem to fit this application into IPR. Experiments on several tasks are also described.

**Chapter 4** is concerned about a *special* IPR-related application, Interactive Text Generation (ITG). In this case, no input pattern is available and the system can only rely on the user feedback. The aim is to provide a system able to semi-automatically generate text documents. Different theoretical and practical issues will be addressed and experiments on very different tasks will be discussed in detail.

**Chapter 5** is focused on an important IPR issue, multi-modality and how to deal with the user feedback. Multi-modal interfaces can be naturally included into IPR and the very nature of the IPR can facilitate their development. A specific multi-modal computer assisted translation system will be studied in this chapter from both theoretical and practical points of view.

Next, in **Chapter 6**, several conclusions about the work developed are presented. In addition, different future lines or problems suitable to be addressed are proposed.

Finally, the bibliography used in the work is enumerated along with an appendix describing real prototypes for the applications proposed here.

# CONTENTS

CHAPTER 1

# INTRODUCTION

## 1.1 Pattern Recognition

Pattern Recognition (PR) can be defined as "the act of taking in raw data and taking an action based on the category of the pattern" [16]. PR is heavily related to perception and, therefore, the most straightforward application of PR is the design and construction of systems able to imitate (to some extent) the human senses. The benefits obtained by the application of such systems are clear and huge. On the one hand, there are environments where using human beings is not possible or it is too risky (the outer space, the ocean depths, the inner earth, etc.). On the other, the productivity that can be achieved by a human operator is clearly limited. For that reason, in these extreme environments, or when a high throughput is required, automatic systems seem to be the only solution. In addition to these examples, PR can be useful to better understand how biological systems recognize patterns in nature.
A typical PR process usually consists of three steps:

- **Preprocessing:** A signal or stimulus is captured from the real world. This stimulus can contain a set of patterns to be recognized along with some useless data. In this step, a segmentation process is usually carried out in order to separate the different patterns captured. In addition, the noise carried by the signal is removed or limited.
- **Feature extraction**: Once the input is segmented and *clean*, the relevant information is extracted. The point here is to achieve a suitable representation for the upcoming recognition process.
- **Recognition**: The final step consists in interpreting the input pattern.

From all the ways in which a pattern can be characterized (for instance, by means of a complex linguistic description or by a set of nominal features), labeling the pattern as an instance of a class is, maybe, the most convenient way for an automatic processing. To this end, this label has to somehow summarize the relevant information included into the pattern. This classification can be performed on a previously

defined set of classes (supervised classification) or, alternatively, can be intended to group the patterns into a set of unknown classes that will be discovered during the classification task itself (unsupervised classification).

Nevertheless, regarding the recognition process as a mere classification task can turn out in a very constrained point of view. In the case of Natural Language Processing (NLP), discussed in section1.2, it is more appropriate to consider the final PR step as an interpretation process.

This step can be approached following different techniques. A deductive approximation could be used when the knowledge needed to perform the recognition is available and can be, as well, properly formalized and represented. However, this knowledge is normally not available or it is extremely vague or imprecise and, because of this, inductive techniques are often more appropriate here. These techniques are based on *learning* a model from a set of samples (training samples) that, somehow, captures the information needed to solve the problem. This model attempts to extract general patterns from the training samples in order to recognize future inputs. Statistical PR is one of the most representative examples of this inductive approximation. In this case, the model is actually a (or a set of) probability distribution that relates the possible inputs to the recognition outcomes. Here, each (properly represented) pattern can be seen as a point in a $d$-dimensional space which has to be scored according to this probability distribution. In the case of NLP, a pattern is actually defined by a set of relationships among these points.

Formally, given an input pattern $x$ and a previously trained model $M$, the classifier will produce a recognition hypothesis $\hat{h}$, from the set $\mathcal{H}$ of all the hypotheses considered, leading to the following expression [a]:

$$\hat{h} = \underset{h \in \mathcal{H}}{\text{argmax}}\, Pr_M(h \mid x) \tag{1.1}$$

Eq. (1.1) formalizes the *optimal classification rule* and it is aimed at minimizing the number of recognition errors produced.

Usually, some information about the prior probability of each hypothesis is available and we can benefit from including this information into our statistical model. By applying Bayes' theorem to Eq. (1.1), we can obtain Eq. (1.2).

$$\hat{h} = \underset{h \in \mathcal{H}}{\text{argmax}}\, Pr(x \mid h) \cdot Pr(h) \tag{1.2}$$

Now, we still have a model that connects the inputs to the classes $Pr(x \mid h)$ but we have, as well, a new term for the hypotheses prior probability $Pr(h)$.

## 1.2 Natural language processing

Natural language processing (NLP) is about the human language. The rationale behind NLP is to provide methods to automatically deal with this kind of language.

[a] Usually, $M$ is assumed to be known and it is not included as part of the notation

Among all the problems addressed by NLP, we can cite:

- Information extraction
- Automatic summarization
- Text-to-speech conversion
- Speech recognition
- Machine translation
- Handwritten text recognition
- Dialogue systems
- etc.

In spite of the fact that human language is the most natural way for us to communicate, it is not clear how to represent and capture its relevant features to allow for an utterly automatic processing. Pattern Recognition can be applied to NLP since, on the one hand, some NLP tasks are actually about pattern decoding (for instance, speech recognition) and, on the other, some PR techniques are well suited to deal with other NLP problems (for instance, machine translation). In NLP, we can generally find an underlying structure within the set of features that describe the input pattern (syntactic pattern recognition). This way, this pattern can be decomposed into simpler sub-patterns and the interpretation process is usually performed by analyzing these sub-patterns and the relationships among them.

In order to make this decomposition, it is usually assumed the language to be a *Markovian* process. Thus, the input is split into sequentially consecutive sub-patterns which are interpreted according to the local structure of the global pattern. In this sense, *Hidden Markov Models*, used in speech recognition to identify the phonetic constituents of the spoken discourse, and $n$-gram language models, widely employed to cope with the syntactic structure of a sentence, will be described in Chapter 3.

Finally, it is interesting to mention that most NLP problems are concerned with transforming an input language fragment into a different representation. Speech recognition, machine translation or automatic summarization are examples of this. Nevertheless, NLP can be also used in generation tasks (for instance within a dialogue system). A particular case of this approach will be studied in this work; namely, an application to automatically suggest portions of text.

## 1.3 Scientific goals

The main scientific goals of this thesis can be summarized in the following points

- Firstly, we try to develop a general framework for Interactive Pattern recognition. This framework should allow for deriving specific applications.
- Secondly, some representative IPR applications should be explored. A formal development along with a detailed experimental work would have to be carried out in order to assess the feasibility of the applications developed.
- Different inherent IPR issues should be studied in detail. In particular, the opportunities arisen from aspects as multi-modality or adaptive learning are interesting points to be considered.

## 1.4 Description of the Tasks used in the experiments

In order to evaluate the different approximations presented, several experiments involving real tasks have been carried out. In this section we are going to give an overview of the tasks used. More details will be given when describing the specific experimental framework for each application.

### 1.4.1 EUTRANS

EUTRANS is a corpus devised during the EUTRANS project [18]. It provides sentences corresponding to the interactions performed in a hotel desk. This corpus is a low perplexity corpus with a vocabulary of about 700 words. On the one hand, the EUTRANS corpus has been used to easily test the different proposals and, on the other, to test if even in tasks where the automatic systems are able to provide high accurate results, the interactive approach can be still useful.

This corpus has been used, specifically, in CAST (Chapter 3). In this case, 335 spoken utterances were employed. Each utterance corresponds to a sentence in Spanish. In addition, the text part of the corpus has been also employed in experiments of Interactive Text Generation (ITG, Chapter 4). About 3000 sentences in Spanish were used as test set.

### 1.4.2 ALBAYZIN geographic corpus

*Albayzin* [15] is a corpus designed to facilitate the development of automatic speech recognition systems for Spanish. Three corpora were built with different purposes (acoustic model training, problematic environments and a real application testing). One of these corpus consists of sentences corresponding to oral queries to a geographic database. About 1500 oral sentences were employed to test CAST (Chapter 3). In addition the text transcriptions were used in ITG (Chapter 4) as an example of a real application: accessing to a database by means of natural language.

### 1.4.3 XEROX corpus

The *Xerox* corpus was produced during the TransType 2 project [13], as a realistic task to fully test the Computer Assisted Translation (CAT) proposal. This is a parallel corpus about printer manuals from the *Xerox* company. This task is more difficult than the two previously described because both the size of vocabulary and the perplexity are significantly higher (about ten thousand words and about six times higher, respectively).

From this task, two corpora were derived. A corpus consisting of whole-spoken sentences was used for testing CAST. The corresponding transcriptions were also used in ITG as a good opportunity to compare the accuracy of ITG and CAST. The second corpus was specifically designed to test the multi-modal interface for CAT systems described in Chapter 5 and it is composed of sentence fragment utterances. These fragments are aimed at selecting parts of the system predictions and/or dictating possible continuations to a previously validated prefix.

### 1.4.4 WSJ corpus

The Wall Street Journal speech corpus [42] is a corpus widely adopted in speech recognition experiments. This corpus is actually split into two different task with vocabularies of 5000 and 20000 words respectively. In this work, the WSJ corpus was used to test the different wordgraph based approximation to CAST. (Chapter 3).

### 1.4.5 Other ITG tasks

ITG was also assessed on additional corpora. First, the EUROPARL Corpus, containing transcriptions of parliamentary sessions. This corpus is considered a difficult task owing to the vocabulary size and perplexity and constitutes a good example to see how ITG can behave in a very realistic situation. On the other hand, a very different task was also adopted. Several *Shakespeare's* plays were collected from publicly available web sites to check ITG in an extreme environment were few samples are available for training purposes and the test vocabulary is very different from the training one. Finally, some of the source files of the *GNU Linux* operative system were also employed as a new task not so much related with NLP.

CHAPTER 2

# INTERACTIVE PATTERN RECOGNITION

## 2.1 Motivation

Nowadays, Pattern Recognition systems are widely used. Speech recognition, for instance, is employed in very different environments. Phone customer services, desktop dictation software, biometrics, are only a few examples. This is also true for other tasks as machine translation, image recognition, etc. Nevertheless, these systems are not perfect and, therefore, some amount of errors will be produced. This is acceptable in some situations, where the benefit obtained by the use of such systems makes up for imperfect results.

Conversely, in some scenarios, it may not be convenient to rely on automatic Pattern Recognition. We can focus, for instance, on the case of machine translation. When translating legal documents, a small change in one term could cause a complete change in the global meaning with all the problems that it would entail. The same thing happens in fields such as medicine, technical manuals for critical systems in engineering, etc. When a perfect outcome is needed, the presence of a human operator is required to verify and correct the system results. This human post-processing is reasonable when the amount of mistakes is not too high and they are, as well, easy to correct. Otherwise, a completely manual process can be more adequate.

In addition to these examples, there are environments where the complete automation does not make any sense. As an example, we can cite one of the problems addressed in this thesis. The basic idea consists in helping a user to generate text documents by reducing typing effort. We could think how important this effort-reduction could be for a disabled person that has to communicate with his/her environment by typing text through specific devices. Here, a fully automatic system is not possible since we need some user feedback (in the form of the previously produced text, as will be discussed later) to be able to generate new responses.

## 2.2 Introduction to Interactive Pattern Recognition

Usually, Pattern Recognition systems are built based only on the kind of inputs expected and on the outputs to be produced. The role of the human operator (if even considered) is rarely included into the system itself. Instead, this is merely regarded as a minor implementation issue and, the treatment of the system outcomes is not something to be concerned about.

In this chapter, we aim at developing a framework where the user actively takes part in the process. As a result, we will have a semiautomatic and interactive system. We call this Interactive Pattern Recognition (IPR)[52].

Before continuing, the benefits that we can expect from this kind of approach should be discussed. As the most relevant points, we can cite:

- The cooperation between an automatic Pattern Recognition system and a human user ensures the achievement of a perfect result (this is guaranteed since the user is who actually controls all the process). On the other hand, the throughput can be significantly increased (the automatic part of the system provides this feature) in comparison to a whole manual process.

- The adoption of an interactive paradigm should improve the system ergonomics. The user is now part of the system and he or she is not limited to deal with the final (and usually imperfect) system outcomes. Besides, the system suggestions can help the user to consider new solutions to the problems being solved. For instance, in a translation task, the system could achieve alternatives that the human translator was not thinking of at that moment.

- The system can benefit from the fact that the user is constantly providing validation and/or corrections to its outputs. This feedback can be used to improve the accuracy in several ways (see below). Typically, a Pattern Recognition system does not have the opportunity to check whether its responses were right or not and, what is more, to obtain a corrected version for the wrong ones. In IPR, on the contrary, the system is always aware of the correct answers for the previous inputs.

In Figure 2.1 a possible architecture for an IPR system is depicted. The IPR operation mode is summarized in the following steps:

1. Initially, when a new input pattern is available, the system proposes a initial hypothesis for this input. In this case, it actually behaves as a typical and automatic Pattern Recognition system.

2. The hypothesis produced is shown to the user, who starts a validation process. If an error is found, some feedback about this error is sent to the system. Otherwise, if the prediction is fully correct, the process is finished and the current hypothesis constitutes the final result (the fact that the user completely validates an outcome can be also considered as useful feedback information).

3. The system benefits from the previous feedback to obtain a new (and hopefully improved) prediction.

4. Go to step 2.

As can be observed, the process is completely human-supervised. The basic goal is to take advantage of the user interactions to produce better and better hypotheses until a perfect one is achieved.

The previous discussion is focused on the opportunities that the available user feedback brings in the short term (specifically, in the decoding of the current input). Nevertheless, as will be discussed in section 2.5, we can also take advantage of this cooperation in the long run.

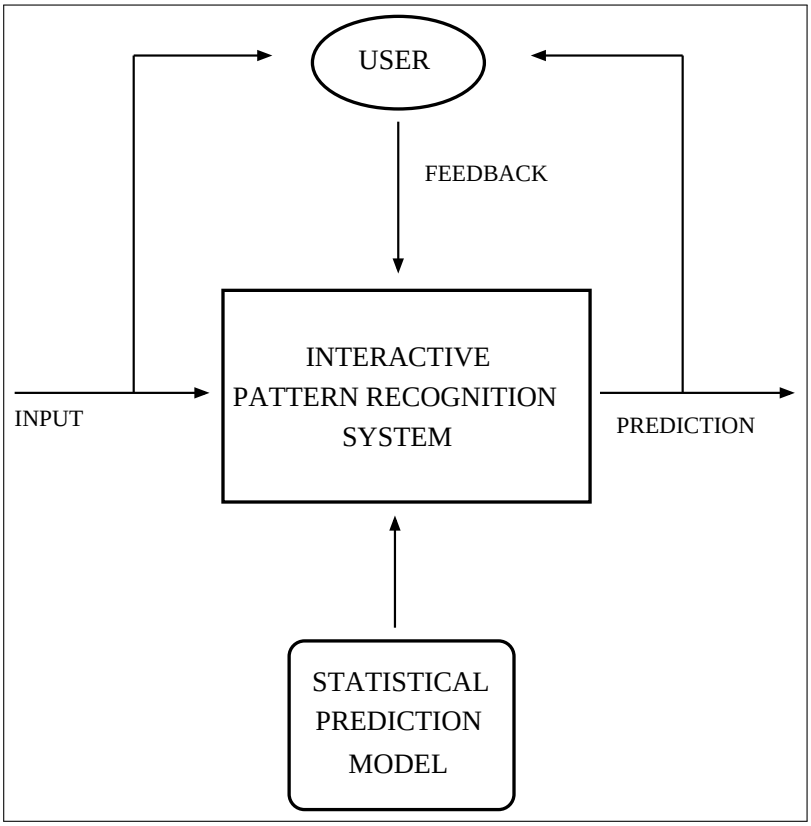



**Figure 2.1:** Architecture of an IPR system.

## 2.3 A formal framework for Interactive Pattern Recognition

In this section we present a formal framework for IPR. Let $x \in \mathcal{X}$ be an input observation or signal (i.e. the pattern to be recognized). Let $h \in \mathcal{H}$ be a hypothesis or prediction (a recognition outcome), derived by the system from $x$ and let $f \in \mathcal{F}$ the feedback coming from the user.

When $x$ arrives to the system, an initial prediction $h$ is generated (no user intervention yet). In this situation, the *classical* approach to Pattern Recognition discussed in Chapter 1 is used to derive this first hypothesis:

$$\hat{h} = \underset{h \in \mathcal{H}}{\operatorname{argmax}} \Pr(h \mid x) \tag{2.1}$$

Once this initial outcome is produced, the user, after analyzing $x$ and $\hat{h}$, sends some feedback $f$ to the system. From now on (i.e. in the second and successive predictions), we will incorporate this feedback into the recognition process trying to make the following hypotheses more accurate.

$$\hat{h} = \underset{h \in \mathcal{H}}{\operatorname{argmax}} \Pr(h \mid x, f) \tag{2.2}$$

where $f$ stands for the feedback, interaction-derived informations; e.g., in the form of *partial hypotheses* or *constraints* on $\mathcal{H}$.

Clearly, the more feedback available the greater the opportunity to obtain better $\hat{h}$. Nevertheless, solving the maximization (2.2) may be more difficult than in the case of our Eq (2.1). Adequate solutions will be discussed when presenting specific applications for this framework in the next chapters.

A direct estimation of this model might not be feasible. From here, and applying Bayes' rule, we can write:

$$\hat{h} = \underset{h \in \mathcal{H}}{\operatorname{argmax}} \Pr(x \mid h, f) \cdot \Pr(f \mid h) \cdot \Pr(h) \tag{2.3}$$

which allows us to introduce two new statistical models to deal with the problem and, therefore, to (hopefully) perform a more reliable estimation. On the one hand, $\Pr(f \mid h)$, accounts for the probability of observing a specific feedback action given the current hypothesis and, on the other, $\Pr(h)$ models the hypothesis prior probability.

In the case of NLP and, owing to the sequential nature of language, the following strategy seems to be quite natural. Firstly, the system provides a whole prediction for the input and the user reads it sequentially. When an error is found, the prefix coming before the mistake is preserved and the rest of the sentence is removed. The current prefix implicitly includes the sequence of the previous interactions, and, therefore, they are not needed anymore. As a result, just the last interaction is really useful. An example of this, borrowed from Computer Assisted Translation (CAT), can be seen in Figure 2.2.

| | | |
|---|---|---|
| **ITER-0** | ($\mathbf{t}_p$) | ( ) |
| **ITER-1** | ($\hat{\mathbf{t}}_s$) | (*Haga clic para cerrar el diálogo de impresión*) |
| | (a) | (Haga clic) |
| | (k) | (**en**) |
| | ($\mathbf{t}_p$) | (Haga clic en) |
| **ITER-2** | ($\hat{\mathbf{t}}_s$) | (ACEPTAR *para cerrar el diálogo de impresión*) |
| | (a) | (ACEPTAR *para cerrar el*) |
| | (k) | (**cuadro**) |
| | ($\mathbf{t}_p$) | (Haga clic en ACEPTAR para cerrar el cuadro) |
| **FINAL** | ($\hat{\mathbf{t}}_s$) | (*de diálogo de impresión*) |
| | (a) | (*de diálogo de impresión*) |
| | (k) | (#) |
| | ($\mathbf{t}_p \equiv \mathbf{t}$) | (Haga clic en ACEPTAR para cerrar el cuadro de diálogo de impresión) |

**Figure 2.2:** Example of a Computer Assisted Translation Session to translate the sentence *"Click Ok to close print dialog"*. In the first interaction the prefix $t_p$ is empty (no user feedback is available yet) and the system produces a whole translation $\hat{t_s}$. Next, the user set a prefix ($a$) of this translation. Then, he or she adds some text $k$, thereby generating a new translation prefix ($t_p$). This process is iterated until the user validates the whole system suggestion.

Clearly, other alternatives are possible. For instance, let's consider an application to recover images from a database. The system provides the user with a set of possible candidates and the user can label some as "appropriate" and some as "inappropriate". For a specific interaction, the current feedback would consist in both set of images (the "appropriate" and the "inappropriate" ones). To take full advantage of this user feedback, not only the last set of labeled images is useful but also the previous ones. Because of this, the whole feedback history is actually informative and the approach described in Eq. (2.2) would have to be strictly followed.

Although we claimed that, for NLP applications, the first order approach turns out to be quite useful, there are, indeed, some opportunities to be explored by tracking all the past user actions. We could, for instance, identify recurrent system mistakes which are, usually, really annoying in order to try to prevent these errors from occurring in the future.

## 2.4 Multi-modality in IPR

One of the consequences of the previous IPR description is that multi-modality appears in a very natural way since the system has to deal with two kind of inputs. On the one hand, we have the input pattern ($x$) to be decoded and, on the other, we have a set of user feedback actions ($f$). The domains from where both inputs come are often very different. In the CAT example, the input is a text source sentence. Regarding the user actions, typing text can be used to introduce new amendments (no

interesting multi-modality yet) but we could also allow for more natural communication modalities. Speech or an e-pen are some alternatives to consider. Actually, several interaction modalities can be allowed simultaneously, which turns out in a multi-modal interface for an IPR.

The description above could be seen as a rough attempt to unnaturally introduce multi-modality here. However, the very nature of IPR systems can really boost the development of multi-modal interfaces. In CAT, we have proposed the use of speech to dictate the corrections [51]. The point here is that we expect the user to utter words that are translations of part of the source sentence. This way, we could use this knowledge to improve the speech recognition accuracy. On the other hand, letting the user choose among different input modalities should make the system more comfortable, thereby increasing the final throughput.

The previous example illustrates how a multi-modal interface can be naturally devised within IPR. Generally speaking, this multi-modal interface should be intended to decode the possible user feedback actions (coming from different domains: speech, keyboard, mouse, e-pen, gestures, etc.) into a suitable representation for the system. This multi-modal interface can benefit from the IPR environment by taking advantage of the available information (input pattern to be recognized, IPR current hypothesis, etc.) in order to better interpret the different user actions. In Figure 2.3 a possible structure for multi-modal IPR is depicted.

From a more formal point of view, we have the system hypothesis $h$, the input pattern $x$ and the current user action, $a$. The goal is to decode $x$ into a proper feedback information $\hat{f}$ for the IPR system:

$$\hat{f} = \operatorname*{argmax}_{f} \Pr(f \mid x, h, a) \tag{2.4}$$

Now, some assumptions can be made leading to different scenarios. This general framework for multi-modal IPR will be discussed for a specific speech and text input interface in Chapter 5.

## 2.5 Adaptive learning

Under the IPR paradigm, we are always exposed to the user feedback. So far, this fact has been only used to improve the accuracy in the short term (for a given input pattern) but this could be also useful to increase the system accuracy in the long term (that is, for future inputs). Maybe, the simplest way, but not the only one, consists in considering the (perfect) system outcomes as new training material as it is described next.

When developing a statistical Pattern Recognition system, two well-separated stages are usually identified. In the first place, a statistical model is learnt from a set of samples or training set. As a result, we obtain a probabilistic model that, to some extent, captures some general knowledge from these samples. At this point, the system can begin its normal operation mode, recognizing new samples. This is usu-

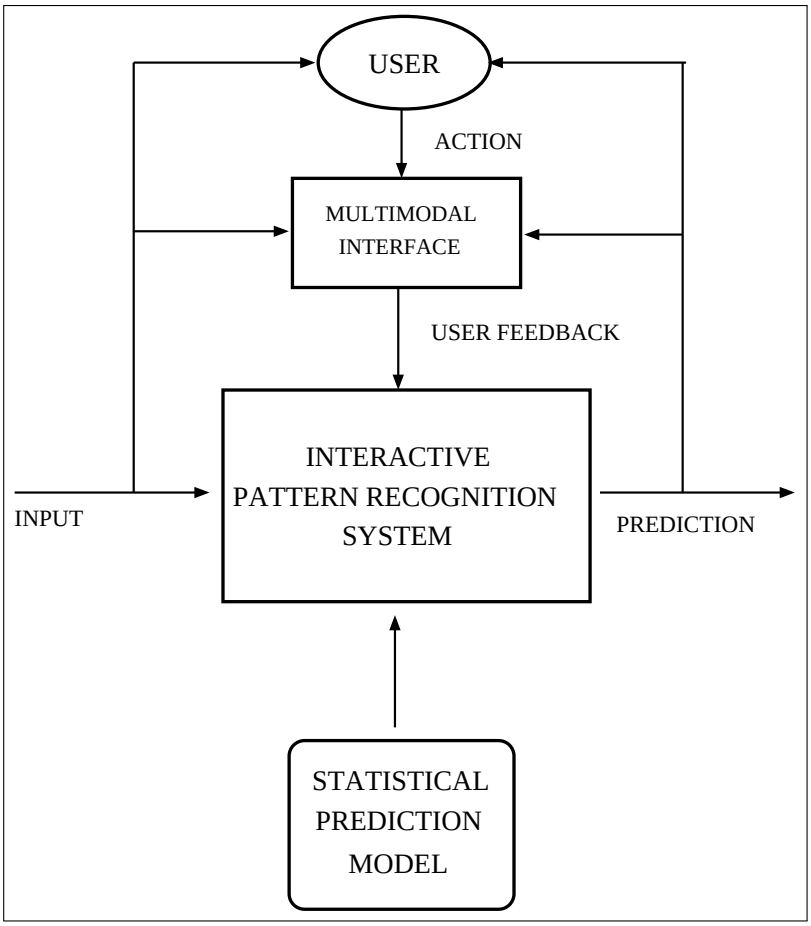



**Figure 2.3:** Multi-modal IPR system

ally known as “batch-training”. In IPR, a similar strategy can be followed. However, we have a user that ensures the correctness of the results produced during the operational stage and, thus, they can be used to improve the statistical prediction models. Moreover, we can expect to find some similarities in the inputs processed in a given moment. This way, the knowledge from the current input (and the corresponding outcome) will be quite useful for the next ones. In other words, the opportunity of continuously adapting the system to the task being currently solved easily arises within IPR. In Figure 2.4 a possible architecture for IPR incorporating adaptive learning is shown.

It can be argued that a similar approach can be followed in *classical* Pattern Recognition, where the system outputs can be also used as new training material. However, this could be a double-edge sword unless we have high confidence in our system accuracy. Besides, we have to tackle the issue of dealing with new events not observed before, In IPR, the user can, implicitly, introduce new elements not seen in

the training stage. For instance, in NLP, the user can produce a so far unknown word. Following an IPR approach, this new word can be easily incorporated into a language model. In fully automated Pattern Recognition, to the best of our knowledge, this can not be done reliably.

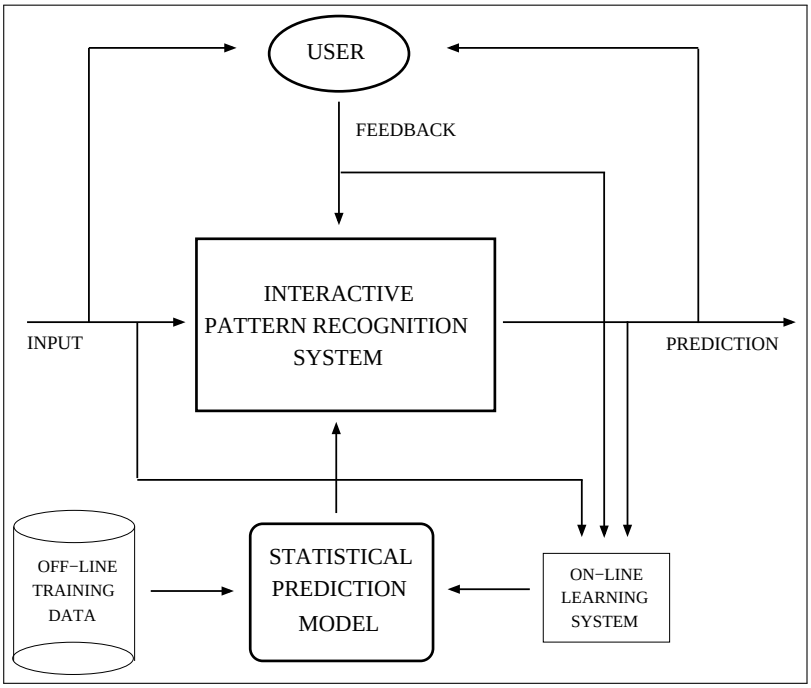



**Figure 2.4:** IPR system incorporating an adaptive learning module.

## 2.6 Evaluating an IPR System

In Pattern Recognition, performance is usually measured by considering the ratio between the number of times that an input has been incorrectly recognized and the overall number of inputs processed. In the specific case of NLP, more sophisticated metrics are adopted. We can cite *Word Error Rate* (WER), *Character Error Rate* (CER) for speech recognition or *Translation Word Error Rate* (TWER) and BLEU for machine translation. WER and TWER can be seen as an adequate estimation of the off-line, word-level post-edition effort required to achieve a perfect result (i.e, the number of words that have to be inserted, deleted or modified).

For IPR, a different strategy has to be followed. Perfect results are guaranteed and, hence, it is necessary to focus on how the system can increase the user productivity. To this end, we can consider two alternative baseline scenarios; a complete manual process or, instead, a human post-processing the outputs of an automatic system. We can assume the second option to be more realistic in most situations and, from now on, we will adopt it as our baseline. Notice that using the first scenario

would generally translate into better results for our proposals but, this could lead us to achieve biased and less accurate conclusions.

From this point of view, an IPR system should perform better when the human intervention is low (the lower the user activity is, the quicker the results are obtained). Thus, a possible way to assess the IPR performance is based on measuring the user effort needed to carry out some predefined task. Several metrics have been developed to this end. On the one hand, *Word Stroke Ratio* (WSR) estimates the average number of word-level user interactions actually performed. On the other hand, *Key Stroke Ratio* (KSR) defines an interaction as a key stroke, this way measuring the average number of user key strokes. Despite what the previous description can suggest, real users do not have to be involved in the experiments, since they can be simulated by means of a suitable set of reference results (that are usually human-generated, though). We can go now a little bit further to conclude that it is possible to estimate the amount of effort that an IPR system can save (or, equivalently, the increase in productivity that can be expected) with respect to a fully automatic system plus a post-edition process by comparing the off-line measures (WER, TWER or CER) to the online (and interactive) ones (WSR or KSR). The difference in the figures can be then used as an estimator for this effort reduction.

At this point, we can not elude an important issue. We really think that real users are necessary to get a clear picture of the actual performance and, hence, they will have to be eventually included into the experimental framework. The evaluation method can be improved by means of other quality indicators obtained from humans performing real sessions with the system. However, in this work we try, as much as we can, to follow a strictly scientific point of view. Because of this, we need some objective way to analyze the techniques that will be proposed in the following chapters and, therefore, WSR and KSR will be employed all along this work.

Finally, we would like to introduce a final consideration. Automatic or semi-automatic systems are not always welcome in real situations. Some people are reluctant to rely on these assistance tools since they can feel they are not actually *in charge* of the process. This is especially noticeable in tasks entailing some sort of creative work. Translation is good example of it. A human translator usually likes to leave his or her mark on the final result. IPR is not planned to interfere with the user work but only as a tool to speed the process up. The user has always the last word.

## 2.7 Summary of contributions

In this first chapter, a formal framework for developing IPR applications has been described in detail. Next, different architectures for IPR systems has been also discussed. Finally, we have proposed some extensions to the initial core of IPR, based on the inclusion of two interesting features: multi-modality and adaptive learning. All these contributions extend the basic IPR proposal presented in [52].

CHAPTER 3

# INTERACTIVE SPEECH RECOGNITION

## 3.1 Introduction to Speech Recognition

One of the most interesting and successful applications of Pattern Recognition is *Automatic Speech Recognition, (ASR)*. Human computer interaction, dialogue systems or transcription of speeches, are only a few examples of the use of ASR in everyday life. However, ASR systems are not perfect. Some of the issues that make the ASR problem hard to be solved are:

- Speaker variability. The speaking style is usually not uniform among people. Features such as voice, accent, cadence, etc. are highly variable and cause confusion to ASR systems. In addition, the language (vocabulary, grammatical constructions, etc) is also speaker-dependent.
- Spontaneous speech. Spontaneous speech is considerably more difficult (pauses, corrections, skipping phonemes or words, etc.) which introduces numerous errors.
- Noise: The signal coming to the system not only contains the speech to be decoded but also additional components (other speakers, environmental noise, etc.). Separating the voice from the rest of the elements in the signal is a challenging task (blind signal separation).

### 3.1.1 Automatic Speech Recognition Systems

An automatic speech recognition (ASR) system takes an input audio signal and decodes this signal producing a text transcription of the words uttered. We will start by describing all the stages needed to achieve this goal.

### Speech acquisition

The human voice generates a series of variations in the air pressure that are transmitted through the air. These pressure changes can be captured by using a special type of transducer (microphone). As a result, this transducer produces an analog electric signal suitable to be stored and processed. However, analog processing presents important drawbacks (noise, need of specific hardware, etc). Computers, on the other hand, are digital systems unable to directly deal with analog inputs. Hence, this signal is converted into the digital domain. In this process, the analog input is periodically sampled and a set of discrete samples is produced as a result. The sampling frequency (that is, the number of samples taken per second) is crucial to ensure an accurate codification of the original signal. According to the Nyquist-Shannon theorem [37] the sampling frequency must be, at least, two times the maximum frequency in the signal. Otherwise, it is not possible to obtain a perfect representation. The maximum frequencies present in a speech signal are around 8 Khz and, therefore, a sampling frequency of 16 Khz is typically used.

### Pre-process and feature extraction

Once the signal is in the digital domain, the relevant information for speech recognition has to be extracted. Different representations have been proposed for speech signals. One of the most widely employed in ASR is based on the use of the so-called *Mel Frequency Cepstrum Coefficients* (MFCCs). These coefficients are obtained as follows. Initially, the signal is split into a sequence of overlapped fragments ("windows" or "frames") where each fragment of signal can be considered a stationary process (each window typically has a size between 10 and 20 milliseconds). The spectrum of every window is then computed and frequencies are grouped into a (non-linear) series of bands (collectively known as filter bank) according to the Mel scale, which is, approximately, linear below 1 KHz and logarithmic above. This way, each speech window (frame) is represented as a vector storing the average of the energy of the frame after passing through the corresponding filter (usually from 20 to 40 filters are employed). Finally, the Discrete Cosine Transform (DCT) is applied to each output vector and the first DCT components (usually from 10 to 15) are chosen. The first and second time-derivative of each DCT vector are usually computed as well.

As a result of this process, the signal is represented as a sequence of feature vectors of dimension between 30 and 40.

### Statistical Speech Recognition

Now that we have the input signal properly pre-processed and represented as a sequence $\mathbf{x}$ (in Figure 3.1 the notation followed in this chapter is summarized) of feature vectors, we can discuss the recognition process itself. In statistical ASR, given an input signal $\mathbf{x}$, we have to obtain the optimal sequence of uttered words $\mathbf{w}$ as it is stated in Eq. (3.1) (notice that, here, $\mathbf{x}$ and $\mathbf{w}$ represent $x$ and $h$ in Eq. (1.1)).

- A math font letter indicates an event ($k$ is the observation in a Markov model, $w$ is a generic word, etc.). This is also used (along with greek letters) to denote a parameter in a model.
- A boldfaced letter indicates a sequence of events ($\mathbf{w}$ indicates a sequence of words).
- A subscripted boldfaced letter denotes an element of a sequence ($\mathbf{w}_2$ indicates the second element of the sequence $\mathbf{w}$).
- A subscripted and superscripted bolfaced letter denotes a subsequence ($\mathbf{w}_1^3$ denotes the first three elements in $\mathbf{w}$).
- $\Pr(\,)$ will be used to denote "true" probability functions, while $P(\,)$ will denote model approximations.

**Figure 3.1:** Summary of the notation used in this and following chapters

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} \Pr(\mathbf{w} \mid \mathbf{x}) \tag{3.1}$$

The first problem to be addressed here is how to estimate this posterior probability. From a practical point of view, we have a set of training examples, where each sample is composed of a speech signal and its corresponding transcription and, from this, we have to perform a reliable estimation of the probability $\Pr(\mathbf{w} \mid \mathbf{x})$. Ideally, we could assume that our set of samples is fully representative of the real probability distribution. However, this assumption is far from being true and, thus, trying a direct estimation of the previous probability is not actually reasonable. As in Eq. (1.2), we can apply Bayes' theorem to Eq. (3.1) to achieve Eq. (3.2).

$$\hat{\mathbf{x}} = \underset{\mathbf{w}}{\operatorname{argmax}} \Pr(\mathbf{x} \mid \mathbf{w}) \Pr(\mathbf{w}) \tag{3.2}$$

In this case, we have two different models that can be estimated separately. As we will see later on, the estimation of $\Pr(\mathbf{x} \mid \mathbf{w})$ is easier than the estimation of the probability in Eq. (3.1) and, on the other hand, the additional term in the maximization, $\Pr(\mathbf{w})$ provides additional information about the hypotheses produced. Specifically, the first term $\Pr(\mathbf{x} \mid \mathbf{w})$ corresponds to an *acoustic model*, which accounts for the distribution of the sounds present in the signal given the set of phonemes in the language.

The second term $\Pr(\mathbf{w})$ is called *language model* and deals with the distribution of the sentences in the language so that correct sentences in the language are (hopefully) scored with high probability and, consequently, incorrect sentences are scored

with low probability.[a]

### 3.1.2 Acoustic modelling

In this section, we are going to address the problem of estimating $Pr(\mathbf{x} \mid \mathbf{w})$. From a generative point of view, we need a model able to produce a speech signal for a given sequence of words with certain probability. Speech is a process in time which means that this model has to be able to deal with temporal series. Moreover, speech can be seen as a non-stationary process described by a set of short-time stationary events. From this, and assuming the Markov property, we can reach a suitable formalism to deal with the acoustic modeling problem.

### 3.1.3 Introduction to Hidden Markov Models

The Hidden Markov Model (HMM) is, so far, the most successful paradigm for stochastic modeling of phonetic units. Formally, an HMM is defined by:

- A set of states $Q = 1, 2, \ldots, N$.
- A transition probability distribution over the states:
  $a_{qq'} = P(q|q')$, where $q, q' \in Q$.
- A emission (observation) probability distribution in each state
  $b_q(k) = P(k|q)$, where $k$ is an observation and $q \in Q$.
- An initial state probability distribution $\pi_q = P(q)$, where $q \in Q$.

The observations can be both discrete or continuous. In the last case, a continuous probability density function is employed and, therefore, the observation probability in each state is specified by the parameters of the density function. In Figure 3.2 an example of a discrete HMM is shown.

#### Hidden Markov Model for acoustic modelling

In speech recognition, the acoustic units usually vary from phonemes to words. As a result, an HMM can be used to model an isolated phoneme, contextual units as diphonemes or triphonemes or even whole words. The main features of HMMs in ASR are:

- On the one hand, the structure of these models is usually a *left-to-right* topology. Each state has one transition to itself and another to the following state (no skip transitions are usually used). This structure is shown in Figure 3.3.

[a] Actually, an additional model is needed to properly deal with the probability $\Pr(\mathbf{x} \mid \mathbf{w})$ since the acoustic model only copes with phonemes and $\mathbf{w}$ is a sequence of words. This will be discussed in section 3.1.6.

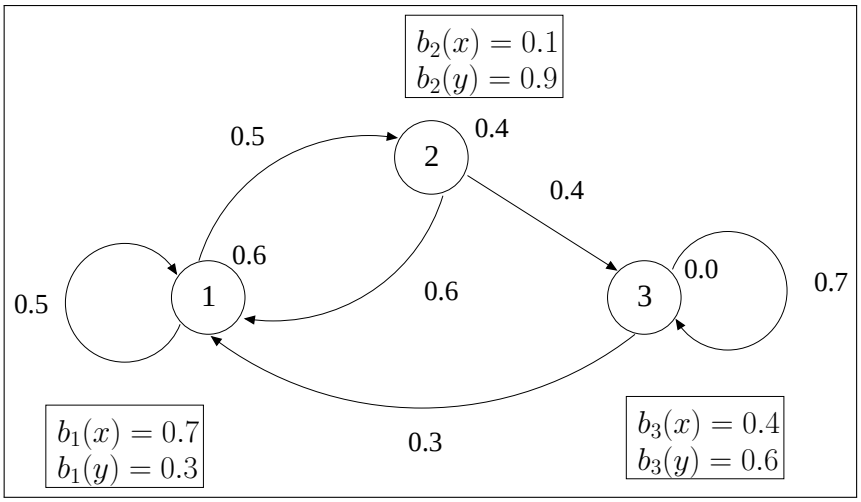



**Figure 3.2:** Example of discrete Hidden Markov Model. This figure depicts a 3 state model where only the symbols $x$ and $y$ can be generated. The probability on the top right of each state represents the probability for this state to be an initial state.

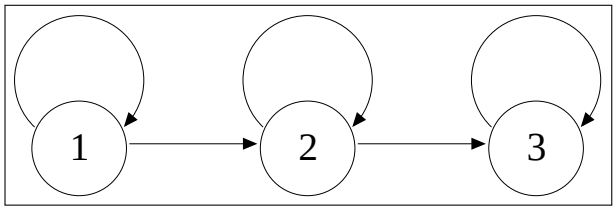



**Figure 3.3:** HMM usual topology for speech recognition

- On the other hand, each observation consists in an n-dimensional vector of continuous components obtained as explained in section 3.1.1. The observation model in each state is a mixture of $M$ Gaussian distributions. Owing to this fact, the probability of generating a vector $k$ in the state $q$ is given by Eq. (3.3), where $c_{qm}$ is the weight (prior) for the $m$-th component in the mixture.

$$b_q(k) = \sum_{m=1}^{M} c_{qm} \cdot \mathcal{N}(k; \mu_{qm}, \Sigma_{qm}) \tag{3.3}$$

### 3.1.4 The estimation problem in HMMs

HMMs are quite useful in speech recognition (as well as in other fields) since the parameters associated to the model can be automatically learned from a set of samples. Actually, the parameters of the probability distributions for both transitions and observations have to be learned.

Approximations based on a Maximum Likelihood criterion have been typically used to this end (however, alternative approaches as, for instance, one based on Maximum Mutual Information [36] can also be found in the literature).

In the case of Maximum Likelihood, the estimation is performed through the *Baum-Welch* or *Forward-Backward* [43] algorithm, which it essentially an Estimation Maximization (EM) algorithm. Before continuing, it is necessary to define the concepts of *Forward* ($\alpha$) and *Backward* ($\beta$) probabilities on which the algorithm relies. In our speech recognition problem, let $T$ be the length of the feature vector sequence $\mathbf{x}$, denoted as $\mathbf{x}_1^T$. The Forward probability is defined as:

$$\alpha_t(q) = P(\mathbf{x}_1^t, q) \tag{3.4}$$

and represents the probability that the HMM is in state $q$ at time $t$, having observed the subsequence $\mathbf{x}_1^t$. This probability can be computed by using the following recursion formula:

$$\alpha_t(q) = \left[\sum_{q' \in Q} \alpha_{t-1}(q') a_{q'q}\right] b_q(\mathbf{x}_t) \tag{3.5}$$

where $\alpha_1(q) = \pi_q b_q(\mathbf{x}_1)$. In our specific ASR three-state topology, we have a single initial state (the first one). So $\pi_q = 1$, if $q = 1$ and $\pi_q = 0$ otherwise. Regarding the Backward probability, it is given by:

$$\beta_t(q) = P(\mathbf{x}_{t+1}^T, q) \tag{3.6}$$

i.e, the probability of generating the sequence $\mathbf{x}_{t+1}^T$ given that the HMM is in state $q$ at time $t$. Again, this probability can be expressed in a recursive way:

$$\beta_t(q) = \left[\sum_{q' \in Q} a_{qq'} b_{q'}(\mathbf{x}_{t+1}) \beta_{t+1}(q')\right] \tag{3.7}$$

where the base case is normally given by $\beta_T(q) = 1/|Q|$. Again, in our specific model, we have a single final state (the third one). So $\beta_T(q) = 1$, if $q = 3$ and $\beta_T(q) = 0$ otherwise.

From the Forward and Backward probabilities, we can estimate the HMM parameters. Specifically, the initial state distribution $\pi_q$, the transition probabilities $a_{qq'}$ and the emission probabilities $b_q(\mathbf{x})$ have to be defined. We employ the notation $\lambda = \{a, b, \pi\}$ to refer to a model and its current parameters [b].

The algorithm starts from an initial set of parameters (this set is often obtained randomly) to iteratively perform a parameter re-estimation to maximize the likelihood of the model given the training samples observed. The re-estimation formulas

[b] The topology of the model, $Q$, is considered previously fixed (see Figure 3.3).

are computed as follows. In the first place, for the transition probability, we can define the probability of being in state $q$ at time $t$ and going to state $q'$ given the current model parameters as:

$$\gamma_t(q,q') = P(q,q'|\mathbf{x}_1^T,\lambda) = \frac{\alpha_{t-1}(q)a_{q,q'}b_q(\mathbf{x}_t)\beta_t(q')}{\sum_{s\in Q}\alpha_T(s)} \tag{3.8}$$

Intuitively, we try to estimate the transition probability $a_{qq'}$ as the expected number of transitions from state $q$ to state $q'$ divided by the overall expected number of transitions leaving state $q$, what leads to the re-estimation formula shown in Eq 3.9.

$$\hat{a}_{qq'} = \frac{\sum_{t=1}^{T}\gamma_t(q,q')}{\sum_{t=1}^{T}\sum_{s\in Q}\gamma_t(s,q)} \tag{3.9}$$

Regarding the observation probability, we define the probability of being in state $q$ at time $t$ as:

$$\gamma_t(q) = P(q|\mathbf{x}_1^T,\lambda) = \frac{\alpha_t(q)\beta_t(q)}{\sum_{j=1}^{T}\alpha_j(q)\beta_j(q)} \tag{3.10}$$

The estimation of the emission probabilities is performed by computing the expected number of times that the process is in state $q$ observing the symbol $k$, divided by the overall expected number of times in state $q$.

In our case, the observation probability is modeled as a Gaussian mixture. Hence, we define the probability of being in state $q$ at time $t$, with the $m$-th component of the mixture accounting for $\mathbf{x}_t$ as:

$$\gamma_t(q,m) = \frac{\alpha_t(q)\beta_t(q)}{\sum_{s\in Q}\alpha_t(s)\beta_t(s)} \cdot \frac{c_{qm}\cdot\mathcal{N}(\mathbf{x}_t;\mu_{qm},\Sigma_{qm})}{\sum_{n=1}^{M}\mathcal{N}(\mathbf{x}_t;\mu_{qn},\Sigma_{qn})} \tag{3.11}$$

From this, we can compute the mixture paramenters as:

$$\hat{c}_{qm} = \frac{\sum_{t=1}^{T}\gamma_t(q,m)}{\sum_{t=1}^{T}\sum_{n=1}^{M}\gamma_t(q,n)} \tag{3.12}$$

$$\hat{\mu}_{qm} = \frac{\sum_{t=1}^{T} \gamma_t(q,m) \cdot \mathbf{x}_t}{\sum_{t=1}^{T} \gamma_t(q,m)} \tag{3.13}$$

$$\hat{\Sigma}_{qm} = \frac{\sum_{t=1}^{T} \gamma_t(q,m) \cdot (\mathbf{x}_t - \mu_{qm})(\mathbf{x}_t - \mu_{qm})^{tr}}{\sum_{t=1}^{T} \gamma_t(q,m)} \tag{3.14}$$

where $^{tr}$ denotes the transposed matrix. Finally, the initial probability distribution is simply estimated according to the number of times in state $q$ at time 1, that is, $\hat{\pi}_q = \gamma_1(q)$ (in our case, this probability has been previously set according to the model topology).

### 3.1.5 The decoding problem in an HMM

After the training stage, the learned HMMs can be used to decode (transcribe) an input speech signal. We can assume, for the sake of simplicity, that each HMM is tied to a phoneme and, therefore, we have as many HMMs as different phonemes exist in a language.

The process of decoding an input utterance consists in finding the most likely sequence of HMMs that can produce this input. As was described before, the forward probability accounts for the probability of a specific model generating a sequence of observations. Similarly, we can devise and algorithm to obtain the best path that reaches a state to finally obtain the best sequence of states and models for an observation sequence. Formally, we can compute the path probabilities $V$ recursively as:

$$\begin{aligned} V_1(q) &= \pi_q b_i(\mathbf{x}_1) \\ V_t(q) &= \max_{q' \in Q} \left[ V_{t-1}(q') a_{q'q} \right] b_q(\mathbf{x}_t) \end{aligned} \tag{3.15}$$

where $V_t(q)$ denotes the probability of the most likely sequence that generates the observations $\mathbf{x}_1^t$ and ends in state $q$. Notice that the sum in the Forward probability has been replaced here by a maximization. Algorithm 1 shows a complete definition of the *Viterbi* algorithm for HMMs decoding.

At this point, we have all the elements needed to decode a speech fragment. Since we known how to obtain the optimal sequence of HMM states that generates an input, we can build a huge HMM , where the final state of each HMM is connected to the initial state of others. By simply computing the most likely state sequence over this huge HMM, the input speech is then represented as a succession of phonetic units.

**Algorithm 1**: Viterbi Algorithm for HMM decoding. Given an observation sequence $\mathbf{x}$ and a HMM $(Q, a, b, \pi)$, the algorithm returns the highest probability sequence of states with which the HMM generates $\mathbf{x}$.

**input** : $\mathbf{x} = \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ The input observation sequence
**output**: The optimal sequence of states for the input observation $\mathbf{x}$
**Algorithm:**
**begin**
  Declare $N = |Q|$
  Declare $V[N+2][T+2]$ and initialize to 0
  Declare *paths*[*numStates*+2][$T+2$]
  **forall** $q \in Q$ **do**
    $V[q][0]=\pi_q$
  $t$=0;
  **while** $t < T$ **do**
    **forall** $q \in Q$ **do**
      **forall** *transition* $q \rightarrow q'$ **do**
        *score* := $V[q][t] \cdot a_{qq'} \cdot b_q(\mathbf{x}_t)$
        **if** *score* $> V[q'][t+1]$ **then**
          $V[q'][t+1]$ := *score*
          *paths*[$q'$][$t+1$] := $q$
    $t := t + 1$
  Perform a backtrace on the $paths$ array starting in the element with highest probability stored in the $t$ column of $V$ and return the resulting path.
**end**

Nevertheless, there are two important issues to be solved. In the first place, the phonetic units have to be transformed into words. Secondly, we have to deal with the second term in Eq. (3.2). In the following sections, these questions will be addressed to obtain, finally, a complete description of a speech recognizer.

### 3.1.6 Lexical modelling

Speech recognition is not merely decoding phonemes. Because written words are composed of letters we need a link between the acoustic units and the letters, syllables and, in the end, words in our language. Hence, some kind of lexical model is needed. In most cases a dictionary storing a word and the corresponding sequence of phonemes is enough. However, more sophisticated models are also possible. For instance, simple deterministic automata are typically employed to permit different pronunciations for a specific word (see Figure 3.4).

Lexical models are normally constructed by following a knowledge-based approach since the information needed to build such models is easily available [55].

house: [haʊs]
cat: [kæt]
table: [teɪbɛl]
... ...

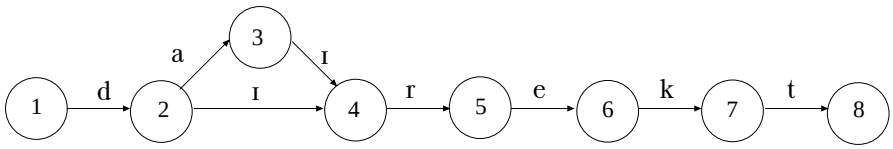



**Figure 3.4:** Example of lexical models. A simple phonetic dictionary can be used to map the words to the phonemes. An alternative is to use a finite state automaton for modelling multiple pronunciations as it is shown for the word "direct"

### 3.1.7 Language modelling

We have already described how to obtain a sequence of words from a speech signal. In very constrained tasks, when only a few spoken commands are used to interact with a computer or to control a robot, for instance, we do not need anything else. Nonetheless, if we want to address more complex tasks, involving decoding whole sentences in natural language, large vocabularies, etc. something more is needed. Acoustic models are not precise enough. Expecting a perfect decoding of all the acoustic units in a sentence is not realistic at all. Things are even worse since we have to tackle situations as disfluent speakers, noisy environments, etc.

When dealing with real NLP applications, it is necessary to use a model that allows us to know whether a specific sequence of words is likely or not to be produced. Given a sequence $\mathbf{w} = \mathbf{w}_1 \dots \mathbf{w}_n$, a statistical language model provides the probability $\Pr(\mathbf{w}) = \Pr(\mathbf{w}_1 \dots \mathbf{w}_n)$ so that a syntactically and semantically correct sequence of words (for instance, "This room is painted in white and gray") is scored with high probability and a bad-formed sequence is given a low probability (for instance, "Black the house is on").

Trying to deal with the joint probability $\Pr(\mathbf{w}_1 \dots \mathbf{w}_l)$ directly is not feasible due to the scarcity of samples usually available. For that reason, an approximation should be taken here. The $n$-gram model is the most widely kind of language model used in ASR (as well as in other NLP tasks). It is based on the following approximate factorization of the joint probability:

$$\Pr(\mathbf{w}_1 \dots \mathbf{w}_l) \approx \prod_{i=1}^{l} P(\mathbf{w}_i | \mathbf{w}_{i-n+1}^{i-1}) \tag{3.16}$$

In an $n$-gram model, each word is conditioned by just the $n-1$ previous words[c].

[c] As in [53], we assume that for any string $\mathbf{z}$, the substring $\mathbf{z}_i^j$ denotes the string $\mathbf{z}_1^j$ if $i \leq 0$ and $\lambda$ if $j \leq 0$. In addition, we assume that $P(\mathbf{z}|\lambda) \equiv P(\mathbf{z})$

In order to allow for an accurate probability estimation, long term dependences are not included in the model. In Figure 3.5 an example of a "2-gram" (bigram) is shown.

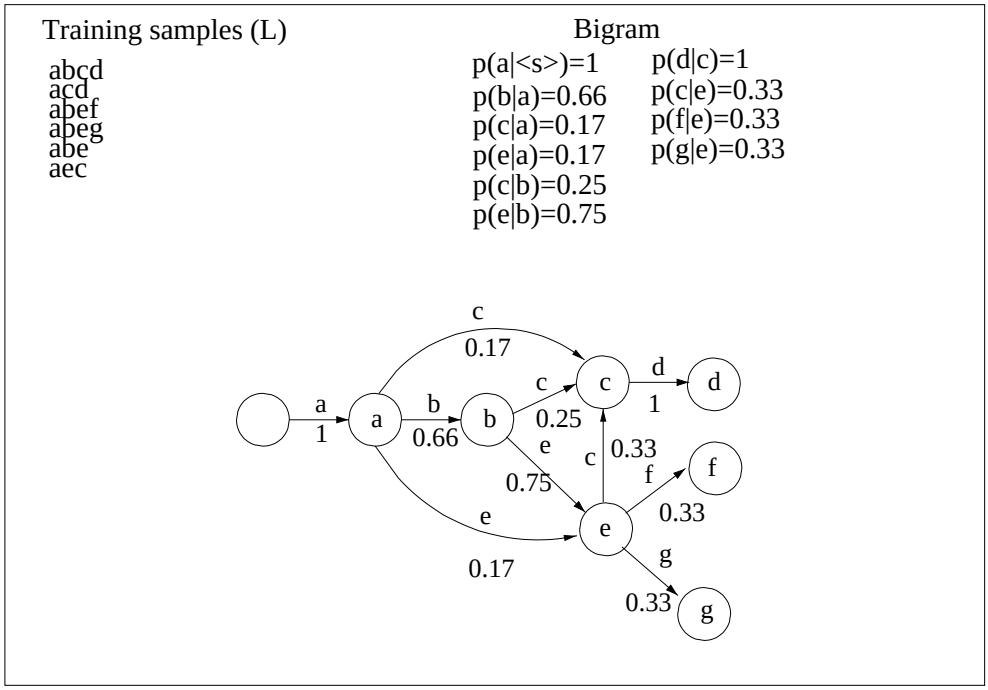



**Figure 3.5:** Example of bigram language model. From the training sample list the conditional probabilities are inferred. This model can also be easily represented as an automaton as the bottom part of the figure shows.

$N$-gram models are estimated by following a maximum likelihood approach. Specifically, the estimation formula is given by:

$$P(w|\mathbf{w}^{(n)})) = \frac{C(\mathbf{w}^{(n)}, w)}{C(\mathbf{w}^{(n)})} \tag{3.17}$$

where $w$ is a word, $\mathbf{w}^{(n)}$ is a sequence of $n$ words, $C(\mathbf{w}^{(n)})$ represents the number of times that the string $\mathbf{w}^n$ occurs in the training set and $C(\mathbf{w}^n, w)$ denotes the number of times that the word $w$ comes after the string $\mathbf{w}^n$ in the training set.

When estimating an $n$-gram model and, due to the scarcity of training data, many of the conditional $n$-gram probabilities are set to zero, independently whether or not they are real strings in the language. This problem has been addressed by proposing an extended version of the $n$-gram model, in which a smoothing of the conditional probability distribution is included. The basic idea is to discount some probability mass from the observed events to be assigned to non-seen events. From all the smoothing techniques that have been proposed in the last years [9], *Kneser-ney* method has been chosen in all the experiments performed in this work.

## 3.2 Computer Assisted Speech Transcription

Once we have described the fundamentals of speech recognition we are going to introduce the Computer Assisted Speech Transcription (CAST) approach [d]. As we discussed in Chapter 2, Pattern Recognition systems are not perfect. In the case of speech recognition, highly accurate results can be obtained in some situations. However, when dealing with complex tasks (especially in the case of spontaneous speech), a significant number of errors can arise. Because of this, when high quality transcriptions are needed, a human transcriber is required to verify and correct the (imperfect) system transcriptions.

As it was also mentioned in Chapter 2, this process is usually performed *off-line*. In the case of ASR, the system initially returns a full transcription of all the available input audio stream. Next, a human transcriber reads it sequentially (while listening to the original audio signal) and corrects the possible mistakes made by the system.

ASR is a good candidate to apply the IPR paradigm. By adopting an interactive scenario, transcriptions could be generated more efficiently. Here, an ASR and a human transcriber can cooperate to achieve a perfect final transcription. This way, we can benefit from the strengths of both contributions (accuracy and productivity).

The CAST operation mode is similar to what was described for generic IPR. The following steps are performed within a CAST session:

1. Initially, for a suitable fragment of the input speech signal, the system proposes a whole initial transcription (in this case, the system behaves like an "standard" ASR)

2. The user goes over the transcription proposed. If no mistakes were found, the process ends with a perfect transcription of the input fragment. Otherwise, the part of the sentence after the last correct word is removed. As a result, we have an error-free prefix.

3. Next, the user adds some words (or characters) to the previous prefix, therefore obtaining a new longer prefix.

4. The prefix generated in step 3 constitutes the user feedback. The system will now complete this prefix by generating a suffix so that a full transcription hypothesis is produced.

5. Go to step 2.

### 3.2.1 Formal framework for CAST

As we discussed in section 3.1.1, statistical speech recognition can be stated as the problem of searching for a sequence of words, $\hat{\mathbf{w}}$, that with maximum probability has produced a given utterance, $\mathbf{x}$ (see Eq. (3.1)).

[d] Speech recognition has a wide range of applications as dialogue, speech translation, human computer interaction, etc. From all these tasks, CAST is intended to work with pre-recorded spoken documents (as parliamentary sessions, lectures, etc.) that require a very accurate transcription.

In CAST, user feedback is available and it can be used to improve the system predictions. In this case, the feedback consists in the prefix $\mathbf{p}$ (notice that here $\mathbf{p}$ represents $f$ in Eq. (2.4)) validated and corrected in the previous iteration. Consequently, the ASR system should try to complete this prefix by searching for the most likely *suffix* $\hat{\mathbf{s}}$ as:

$$\begin{aligned}\hat{\mathbf{s}} &= \operatorname*{argmax}_{\mathbf{s}} \Pr(\mathbf{s} \mid \mathbf{x}, \mathbf{p}) \\ &= \operatorname*{argmax}_{\mathbf{s}} \Pr(\mathbf{x} \mid \mathbf{p}, \mathbf{s}) \cdot \Pr(\mathbf{s} \mid \mathbf{p}) \end{aligned} \tag{3.18}$$

Eq. (3.18) is very similar to Eq. (3.1), where $\mathbf{w}$ is the concatenation of $\mathbf{p}$ and $\mathbf{s}$. The main difference is that here $\mathbf{p}$ is given. Therefore, the search must be performed over all possible suffixes $\mathbf{s}$ of $\mathbf{p}$ and the language model probability $\Pr(\mathbf{s} \mid \mathbf{p})$ must account for the words that can be uttered *after the prefix* $\mathbf{p}$.

In order to solve Eq. (3.18), the signal $\mathbf{x}$ is considered split into two fragments, $\mathbf{x}_1^b$ and $\mathbf{x}_{b+1}^T$, where $T$ is the length in frames of $\mathbf{x}$. By further considering the boundary point $b$ as a hidden variable in Eq. (3.18), we can write:

$$\begin{aligned}\hat{\mathbf{s}} &= \operatorname*{argmax}_{\mathbf{s}} \sum_{0<b\leq T} \Pr(\mathbf{x}, b \,|\, \mathbf{s}, \mathbf{p}) \cdot \Pr(\mathbf{s} \,|\, \mathbf{p}) \\ &= \operatorname*{argmax}_{\mathbf{s}} \sum_{0<b\leq T} \Pr(\mathbf{x}_1^b, \mathbf{x}_{b+1}^T | \mathbf{s}, \mathbf{p}) \cdot \Pr(\mathbf{s} \mid \mathbf{p}) \end{aligned} \tag{3.19}$$

Before continuing, we can make the *naïve* (but realistic) assumption that the probability of the initial signal fragment $\mathbf{x}_1^b$ given $\mathbf{p}$ does not depend on the suffix and the probability of $\mathbf{x}_{b+1}^T$ given $\mathbf{s}$ does not depend on the prefix, to rewrite Eq. (3.19) as:

$$\hat{\mathbf{s}} \approx \operatorname*{argmax}_{\mathbf{s}} \sum_{0<b\leq T} \Pr(\mathbf{x}_1^b \,|\, \mathbf{p}) \cdot \Pr(\mathbf{x}_{b+1}^T \,|\, \mathbf{s}) \cdot \Pr(\mathbf{s} \,|\, \mathbf{p}) \tag{3.20}$$

Finally, the sum over all the possible segmentations can be approximated by the dominating term, leading to:

$$\hat{\mathbf{s}} \approx \operatorname*{argmax}_{\mathbf{s}} \max_{0\leq b\leq T} \Pr(\mathbf{x}_1^b \,|\, \mathbf{p}) \cdot \Pr(\mathbf{x}_{b+1}^T \,|\, \mathbf{s}) \cdot \Pr(\mathbf{s} \,|\, \mathbf{p}) \tag{3.21}$$

This optimization problem entails finding an optimal boundary point, $\hat{b}$, associated with the optimal suffix decoding, $\hat{\mathbf{s}}$. That is, the signal $\mathbf{x}$ is actually split into two segments, $\mathbf{x}_{\mathbf{p}} = \mathbf{x}_1^{\hat{b}}$ and $\mathbf{x}_{\mathbf{s}} = \mathbf{x}_{\hat{b}+1}^T$. The first one corresponds to the *prefix* and the second one to the *suffix*. On account of this, the search for the best suffix can be performed just over segments of the signal corresponding to the possible suffixes and, on the other hand, we can take advantage of the information coming from the prefix to tune the language model constraints modelled by $\Pr(\mathbf{s} \mid \mathbf{p})$. This is discussed in the next subsections.

### 3.2.2 Adapting the language model

Perhaps the simplest way to deal with $\Pr(\mathbf{s} \mid \mathbf{p})$ in Eq. (3.21) is to adapt an $n$-gram language model to cope with the consolidated prefix. Given that a conventional $n$-gram models the probability $\Pr(\mathbf{w})$ (where $\mathbf{w}$ is the concatenation of $\mathbf{p}$ and $\mathbf{s}$, i.e the whole sentence), it is necessary to introduce some modifications to deal with the conditional probability $\Pr(\mathbf{s} \mid \mathbf{p})$.

Let $\mathbf{p} = \mathbf{w}_1^k$ be a consolidated prefix and $\mathbf{s} = \mathbf{w}_{k+1}^l$ be a possible suffix. We can compute $\Pr(\mathbf{s}|\mathbf{p})$ as it is shown in Eq. (3.22).

$$\begin{aligned} \Pr(\mathbf{s} \mid \mathbf{p}) &= \Pr(\mathbf{p}, \mathbf{s}) / \Pr(\mathbf{p}) \\ &\approx \frac{\prod_{i=1}^{l} P(\mathbf{w}_i \mid \mathbf{w}_{i-n+1}^{i-1})}{\prod_{i=1}^{k} P(\mathbf{w}_i \mid \mathbf{w}_{i-n+1}^{i-1})} \\ &= \prod_{i=k+1}^{l} P(\mathbf{w}_i \mid \mathbf{w}_{i-n+1}^{i-1}) \end{aligned} \tag{3.22}$$

Moreover, for the terms from $k+1$ to $k+n-1$ of this factorization, we have additional information coming from the already known words $\mathbf{w}_{k-n+2}^k$, leading to:

$$\begin{aligned} \Pr(\mathbf{s} \mid \mathbf{p}) &\approx \prod_{i=k+1}^{k+n-1} P(\mathbf{w}_i \mid \mathbf{w}_{i-n+1}^{i-1}) \cdot \prod_{i=k+n}^{l} P(\mathbf{w}_i \mid \mathbf{w}_{i-n+1}^{i-1}) \\ &= \prod_{j=1}^{n-1} P(\mathbf{s}_j | \mathbf{p}_{k-n+1}^{k}, \mathbf{s}_1^{j-1}) \cdot \prod_{j=n}^{l-k} P(\mathbf{s}_j | \mathbf{s}_{j-n+1}^{j-1}) \end{aligned} \tag{3.23}$$

The first term accounts for the probability of the $n-1$ words of the suffix, whose probability is conditioned by words from the validated prefix, and the second one is the usual $n$-gram probability for the remaining suffix words.

### 3.2.3 Searching

Once we have a CAST formalization available, a possible implementation of a CATS decoder will be described. In the initial CAST iteration, $\mathbf{p}$ is empty and the decoder has to generate a full transcription of $\mathbf{x}$ as in Eq. (3.1). Afterwards, the user-validated prefix $\mathbf{p}$ has to be used to generate a suitable continuation $\mathbf{s}$ in the following iterations of the interactive process.

A simple possibility would be to perform the decoding in two steps: first, the validated prefix $\mathbf{p}$ could be used to segment the signal $\mathbf{x}$ into $\mathbf{x}_\mathbf{p}$ and $\mathbf{x}_\mathbf{s}$ and, then, $\mathbf{x}_\mathbf{s}$ could be decoded by using a "suffix language model" (SLM) as in Eq. (3.23). The problem here is that the signal can not be optimally segmented into $\mathbf{x}_\mathbf{p}$ and $\mathbf{x}_\mathbf{s}$ if only the information of the prefix $\mathbf{p}$ is considered.

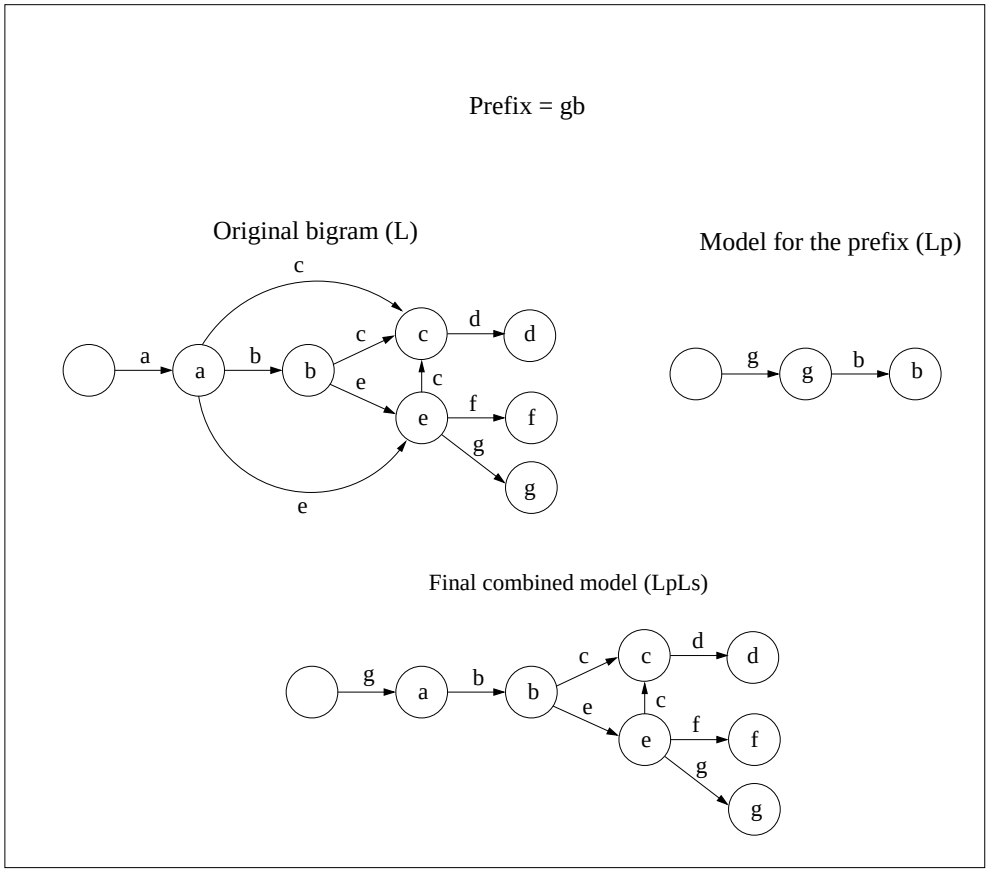



**Figure 3.6:** Example of a CAST language model. Given the $n$-gram language model for whole sentences in the figure a *linear* model ($L_p$) which accounts for the prefix $gb$ is constructed. Then, these two models are combined into a single model ($L_pL_s$) as shown.

A better approach is to explicitly rely on Eq. (3.21) to implement a decoding process in one step, as in *classical* speech recognition. The decoder should be forced to *match* the previously validated prefix $\mathbf{p}$ and then continue searching for a suffix $\hat{\mathbf{s}}$ according to the constraints in Eq. (3.23). To this end, we can build a special language model which can be seen as the "concatenation" of a *linear* model, which strictly accounts for the successive words in $\mathbf{p}$, and the SLM in Eq. (3.23). An example of this LM is shown in Figure 3.6.

Owing to the finite-state nature of this special LM, the search involved in Eq. (3.21) can be efficiently carried out using the same *Viterbi* algorithm [54] as in Eq.(3.15). Apart from the optimal suffix decoding, $\hat{\mathbf{s}}$, a correspondingly optimal segmentation of the speech signal is then obtained as a byproduct.

## 3.3 More Efficient Search Approaches. Using Word Graphs

We have described a possible implementation of CAST that fits well with the formal proposal. However, we have to bear in mind that our final purpose is to develop an application to be employed in a real situation. In spite of the fact that a simple theoretical approach can be useful as a first practical solution, we are going to be concerned, as well, about some important practical issues.

CAST is an interactive application and, as such, some specific requirements have to be fulfilled. For instance, no matter how precise the ASR can be if the time needed to obtain a hypothesis is too high. In the most extreme case, if the prediction system is as slow as the user performing the task manually, CAST does not make any sense. To summarize, we can claim that, in order for the user to feel comfortable with the system, we have to ensure an appropriate time response. Although some experiments in this sense will be described later, we can say, for the time being, that the implementation scheme shown in section 3.2.3 presents a response time higher than *3 seconds* in some tasks. In consequence, we need to explore an alternative CAST implementation.

In a speech decoding, there are many computations to be performed for each frame in the input signal. The acoustic score, for instance, requires to calculate the probability of a Gaussian mixture for each state in all the actives HMMs. We could save a lot computation effort if we were able to obtain, for each input to be transcribed, a representation that stores a sufficient number of decoding hypotheses along with their scores. This way, all the interactive CAST search would be performed on this model, achieving a better time response.

The previous discussion suggests the use of a well known ASR data structure, a word graph. A word graph is, indeed, a compact way to represent a very large set of $n$-best hypotheses along with additional information about how they were produced. Formally, a word graph can be defined as a directed acyclic graph, specified by:

A word graph is an acyclic, directed graph specified by the following parameters:

- A set of nodes $Q = q_1, ..., q_N$.
- A function $t(q)$ that associates a state to a specific time (frame) in the input speech signal.
- A set of arcs $A$, where each arc is defined by $[a_s, a_t, a_w, a_l]$ where $a_s$ denotes the source node, $a_t$ denotes the target node, $a_w$ denotes the word produced by the arc and $a_l$ denotes the likelihood (this likelihood represent the combination of the acoustic and the language model probabilities [e]).

Word graphs can be constructed as a byproduct of the speech decoding process by storing the best acoustic and language model probabilities for each partial hypothesis.

[e] Because acoustic and language model probabilities are expressed in different magnitude orders a *Language Model Scale Factor* is usually employed here to properly combine them.

Next, all the paths starting from initial states and reaching final states are added to the graph [35, 41].

Once the word graph is available for a specific input, it can be used to perform the search described in Eq. (3.21). Now, we will study how to address this search. Basically, the idea consists in, when a new user prefix is available, parsing this prefix over the word graph. This is aimed at obtaining a set of nodes that approximates the best signal segmentation (the first two terms in Eq. (3.21)). Moreover, the prefix-based language model probability in Eq. (3.21) can be easily computed from the arcs leaving these nodes. Once this set of nodes is available, we can produce a CAST hypothesis (suffix) by searching for the best (or the n-best) path starting from these nodes. In the upcoming sections, different details of this process are discussed.

### 3.3.1 Error Correcting Prefix Parsing

In our case, we have a directed acyclic graph and have to find the best path *compatible* with the prefix. Ideally, this graph would contain all the possible recognition outcomes for the input signal but, unfortunately, this is not actually true in practice.

Firstly, the stochastic model that conducts the word graph generation has been trained from a finite set of samples (although smoothed models can be used, there is still the problem of out-of-vocabulary words). Secondly, a pruning search is usually applied because of computer memory constraints. As a result of this, we can not expect the word graph to account for every possible user prefix. For that reason, a more sophisticated approach has to be adopted. This is the case of the Error Correcting Parsing (ECP) described as follows. To start with, we can define an error model to address the problem of generating a string $\mathbf{y} = \mathbf{y}_1, \ldots, \mathbf{y}_n$ from another string $\mathbf{z} = \mathbf{z}_1, \ldots, \mathbf{z}_m$. This generation is based on a well defined set of operations:

- Substitution: Consists in replacing a symbol $\mathbf{y}_i$ in the source string with a symbol $\mathbf{z}_j$ in the target string (denoted as $\mathbf{y}_i \rightarrow \mathbf{z}_j$).
- Deletion: Consists in removing a symbol $\mathbf{y}_i$ in the source string (denoted as $\mathbf{y}_i \rightarrow \lambda$).
- Insertion: Consists in inserting a symbol $\mathbf{z}_j$ in the target string (denoted as $\lambda \rightarrow \mathbf{z}_j$).

Each operation has an associated cost. This cost is usually chosen according to the specific task to be solved. The overall cost of generating one string from another is computed by summing up all the editing costs involved in transforming the source string into the target one. For a given sequence of editing operations $\mathbf{e} = \mathbf{e}_1, \ldots, \mathbf{e}_n$ the cost of $\mathbf{e}$ is then defined as:

$$C(\mathbf{e}) = \sum_{i=1}^{n} cost(\mathbf{e}_i) \tag{3.24}$$

where $cost(\mathbf{e}_i)$ denotes the cost of the editing operation $\mathbf{e}_i$. It is easy to see that a specific target string can be generated from a given source in very different ways. Generally, we are only interested in the sequence of minimum cost. This *optimal* sequence is known as the (weighted) *Levenshtein* distance [46]:

$$d(\mathbf{y}, \mathbf{z}) = \min_{\mathbf{e}} \{cost(\mathbf{e}) \mid \mathbf{y} \longrightarrow^{\mathbf{e}} \mathbf{z}\} \tag{3.25}$$

where $\mathbf{y} \longrightarrow^{\mathbf{e}} \mathbf{z}$ denotes a sequence of edition operations to reach $\mathbf{z}$ from $\mathbf{y}$. To compute the *Levenshtein* distance in a polynomic time, the following dynamic programming algorithm can be followed (notice that $i$ and $j$ denote positions in the source and the target sentence respectively). Given two strings $\mathbf{y}$ and $\mathbf{z}$, $d(\mathbf{y}, \mathbf{z})$ is computed as:

- Recursive general term:

$$\begin{aligned} d(i,j) = \quad \min \Big\{ & d(i-1, j-1) + cost(\mathbf{y}_i \rightarrow \mathbf{z}_j), \\ & d(i-1, j) + cost(\mathbf{y}_i \rightarrow \lambda), \\ & d(i, j-1) + cost(\lambda \rightarrow \mathbf{z}_j)) \Big\} \end{aligned}$$

- Base case:

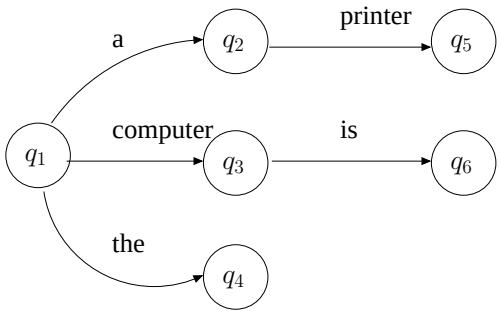
$$\begin{aligned} d(0,0) &= 0 \\ \forall i \; d(i,0) &= d(i-1,0) + cost(\mathbf{y}_i \rightarrow \lambda) \\ \forall j \; d(0,j) &= d(0,j-1) + cost(\lambda \rightarrow \mathbf{z}_j) \end{aligned}$$

In CAST we have a string (prefix) and a representation of many strings along with their probabilities (word graph) and we have to parse the prefix over this graph. This problem is similar to the problem of finding the minimum distance between a regular language and a given string [20].

This algorithm returns the *Levenshtein* distance along with the graph nodes ("non-terminals") reached after parsing the input string. The search for the best suffix can be then performed by applying a *Viterbi*-like search from these nodes. In figure 3.7 an example of ECP over a word graph is shown.

### 3.3.2 A general model for probabilistic prefix parsing

So far, we have a tool (Error Correcting Parsing) that allows us to perform a CAST search within word graphs. However, there are some issues to be discussed before going on with this approach. On the one hand, it is not clear how to relate the ECP procedure to Eq. (3.20) and Eq. (3.21). On the other, as a result of the ECP, we have a set of states with an associated cost (the ECP cost) and probability (the probability given by the path(s) in the word graph that reaches the state). The question is how




Given the previous word graph fragment, and given $\mathbf{p}$="the printer", we can compute the ECP cost to reach each state. For all the states in this model fragment, we have: cost($q_1$)=2 (2 deletions); cost($q_2$)=2 (1 substitution and 1 deletion); cost ($q_3$)=1 (1 deletion); cost ($q_4$)=1 (1 deletion); cost ($q_5$)=1 (1 substitution); cost ($q_6$)=2 (2 substitutions)

**Figure 3.7:** Example of error prefix correcting parsing on a word graph fragment

to combine these two terms to carry out the search for the suffix as Eq. (3.21) shows. Different heuristics can be applied here. For instance, all the states with minimum cost can be chosen as initial states to continue the search for the optimal suffix. This approach has been used in some word graph-based CAT approaches [11].

To overcome this problem, a new formulation can be attempted in order to properly include ECP into word graph CAST approximations. Starting from Eq. (3.18), we can introduce a hidden variable $q_b$ to represent a possible boundary node between the prefix and the suffix in the word graph:

$$\begin{aligned}
\hat{s} &= \underset{\mathbf{s}}{\operatorname{argmax}} \Pr(\mathbf{s} \mid \mathbf{p}) \cdot \Pr(\mathbf{x} \mid \mathbf{p}, \mathbf{s}) = \\
&= \underset{\mathbf{s}}{\operatorname{argmax}} \Pr(\mathbf{s} \mid \mathbf{p}) \sum_{q_b \in Q} \Pr(\mathbf{x}, q_b \mid \mathbf{p}, \mathbf{s}) = \\
&= \underset{\mathbf{s}}{\operatorname{argmax}} \Pr(\mathbf{s} \mid \mathbf{p}) \sum_{q_b \in Q} \Pr(\mathbf{x} \mid q_b, \mathbf{p}, \mathbf{s}) \cdot \Pr(q_b \mid \mathbf{p}, \mathbf{s}) \qquad (3.26)
\end{aligned}$$

Notice that in Eq. (3.19), Eq. (3.20) and Eq. (3.21) the boundary point $b$ can be directly computed on the input signal. Here that point has to be approximated according to the nodes in the word graphs (which are tied to a specific frame in the input signal). We can make the naive assumption that $\mathbf{x}$ does not depend of $\mathbf{p}$ given $q_b$ to rewrite Eq. (3.18) as:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\operatorname{argmax}} \Pr(\mathbf{s} \mid \mathbf{p}) \sum_{q_b \in Q} \Pr(\mathbf{x} \mid q_b, \mathbf{s}) \cdot \Pr(q_b \mid \mathbf{p}, \mathbf{s})$$

Additionally, we can assume that $q_b$ only depends on the prefix (this issue will be discussed later), leading to:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\operatorname{argmax}} \Pr(\mathbf{s} \mid \mathbf{p}) \sum_{q_b \in Q} \Pr(\mathbf{x} \mid q_b, \mathbf{s}) \cdot \Pr(q_b \mid \mathbf{p})$$

Finally, the usual approximation of the sum by the dominating term can be adopted to obtain:

$$\hat{\mathbf{s}} \approx \underset{\mathbf{s}}{\operatorname{argmax}} \Pr(\mathbf{s} \mid \mathbf{p}) \max_{q_b \in Q} \Pr(\mathbf{x} \mid q_b, \mathbf{s}) \cdot \Pr(q_b \mid \mathbf{p})$$

To properly deal with $\Pr(q_b \mid \mathbf{p})$ it is necessary to define the editing operations in a probabilistic way. This can be easily done by constructing an stochastic automaton representing the string to be parsed (in our case, the prefix) so that the different editing operations can be modeled as groups of arcs in the automaton (see Figure 3.8).

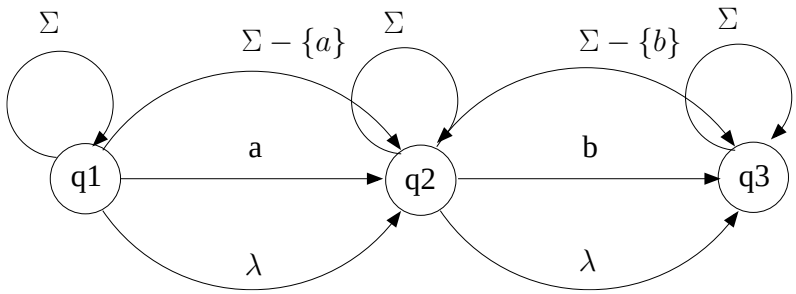



**Figure 3.8:** Example of extended automaton for probabilistic ECP given the prefix $ab$. From each state, we have four groups of arcs. The first one corresponds to the operation of replacing a symbol with itself (arcs labeled as $a$ form q1 to q2 and $b$ from q2 to q3). The second group represents the substitution of a symbol for another symbol. Here, we have an arc for each symbol in the vocabulary except the symbol represented in the previous group. The third group models a deletion, which is represented by the $\lambda$ arc to the next state. Finally, the last group is for insertions, involving an arc for each symbol in the alphabet $\Sigma$ from a state to itself. Notice that, in order to represent a real probability distribution the sum of all the arcs leaving a state must be exactly one.

In the ECP cost-based approach, all the operations are usually defined to have a similar cost except the substitution of a symbol by itself which is usually a no-cost operation. Directly translating these costs into probabilities is not trivial at all. Intuitively, the case of the no-cost could be mapped to a probability of one, since the substitution of a symbol by itself does not entail a real transformation of the string. However, this would imply to use a *null* score for the remaining set of operations. Alternatively, some uncertainty can be assigned to this *special* operation and, this way, some probability mass is available to be distributed among the other ones.

To start with, we can consider any editing operation equivalent. To this end, we can group the probability so that any insertion, deletion and real substitution are

equally likely. This actually means that all the arcs in the ECP automaton in Figure 3.8 should be labeled with the same probability. Those arcs not involving a real transformation of the string, however, will have a different treatment. A much higher probability should be considered for them. By assigning, for instance, half of the overall probability mass to these arcs and equally distributing the other half among the other operations, we can reach the following expressions

$$P(\mathbf{y}_i \rightarrow \mathbf{z}_j) \quad = \quad \begin{cases} \frac{1}{2} & \mathbf{y}_i = \mathbf{z}_j \\ \frac{1}{4|\Sigma|} & \mathbf{y}_i \neq \mathbf{z}_j \\ \frac{1}{4|\Sigma|} & \mathbf{y}_i = \lambda \\ \frac{1}{4|\Sigma|} & \mathbf{z}_j = \lambda \end{cases} \tag{3.27}$$

Here, the score of each editing operation is set so that each arc in the ECP automaton has the same probability (except those arcs that do not transform the input string). $\Sigma$ represents the vocabulary. Notice, that in the case of substitutions and insertions, the probability mass is grouped for all the symbols in the vocabulary (the amount of probability assigned to these groups of arcs would be $\frac{|\Sigma|-1}{4|\Sigma}$ and $\frac{|\Sigma|}{4|\Sigma|}$ for substitutions and insertions respectively) . For that reason, a specific insertion of substitution will be scored with the same probability as a deletion.

Now, we can define $\mathbf{e}_{\mathbf{p}q} = \mathbf{e}_{\mathbf{p}q_1} \dots \mathbf{e}_{\mathbf{p}q_n}$ as a sequence of $n$ editing operations that allows to reach the state $q$ given the current prefix $\mathbf{p}$. Assuming independence among these operations, we can compute this sequence probability as:

$$P(\mathbf{e}_{\mathbf{p}q}) = \prod_{i=0}^{n} P(\mathbf{e}_{\mathbf{p}q_i}) \tag{3.28}$$

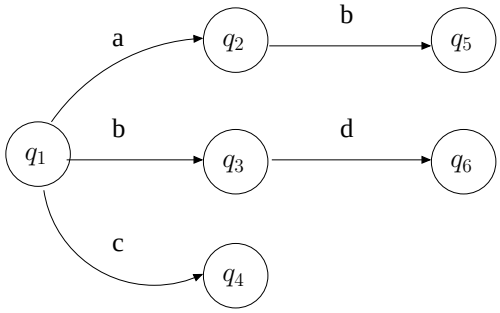
From this, we can easily define the optimal sequence $\hat{\mathbf{e}}_{\mathbf{p}q}$ as:

$$\hat{\mathbf{e}}_{\mathbf{p}q} = \underset{\mathbf{e}_{\mathbf{p}q}}{\text{argmax}}\, P(\mathbf{e}_{\mathbf{p}q}) \tag{3.29}$$

To finally compute the probability $\Pr(q_b \mid \mathbf{p})$ as:

$$\Pr(q_b \mid \mathbf{p}) = \frac{P(\hat{\mathbf{e}}_{\mathbf{p}q_b})}{\sum_{q \in Q} P(\hat{\mathbf{e}}_{\mathbf{p}q})} \tag{3.30}$$

where $Q$ is the set of all states in the word graph. In Figure 3.9, an very simple example of probabilistic error parsing based on this proposal is depicted.




Given the previous word graph, $\mathbf{p}$="$aa$", and $\Sigma = \{a, b, c, d\}$. We can assume, according to Eq. (3.27), the probability for a specific substitution, insertion or deletion as $\frac{1}{16}$. In addition, the probability of substituting a symbol for itself is $\frac{1}{2}$. Therefore, we can compute the ECP probability for each state as:

$$\hat{e}_{\mathbf{p}q_1} = a \to \lambda, a \to \lambda; \qquad \Pr(\hat{e}_{\mathbf{p}q_1}) = 1/16 \cdot 1/16 = 0.0625$$

$$\hat{e}_{\mathbf{p}q_2} = a \to a, a \to \lambda; \qquad \Pr(\hat{e}_{\mathbf{p}q_2}) = 1/2 \cdot 1/16 = 0.0625$$

$$\hat{e}_{\mathbf{p}q_3} = a \to b, a \to \lambda; \qquad \Pr(\hat{e}_{\mathbf{p}q_3}) = 1/16 \cdot 1/16 = 0.0039$$

$$\hat{e}_{\mathbf{p}q_4} = a \to c, a \to \lambda; \qquad \Pr(\hat{e}_{\mathbf{p}q_4}) = 1/16 \cdot 1/16 = 0.0039$$

$$\hat{e}_{\mathbf{p}q_5} = a \to a, a \to b; \qquad \Pr(\hat{e}_{\mathbf{p}q_5}) = 1/2 \cdot 1/16 = 0.0625$$

$$\hat{e}_{\mathbf{p}q_5} = a \to b, a \to d; \qquad \Pr(\hat{e}_{\mathbf{p}q_6}) = 1/16 \cdot 1/16 = 0.0039$$

From this and normalizing on the sum of the previous probabilities ($0.1992$) we can compute the probability $P(q|\mathbf{p}) \ \ \forall q \in Q$ as:

$$P(q_1|\mathbf{p}) = \tfrac{0.0625}{0.1992} = 0.313$$

$$P(q_2|\mathbf{p}) = \tfrac{0.0625}{0.1992} = 0.313$$

$$P(q_3|\mathbf{p}) = \tfrac{0.0039}{0.1992} = 0.020$$

$$P(q_4|\mathbf{p}) = \tfrac{0.0039}{0.1992} = 0.020$$

$$P(q_5|\mathbf{p}) = \tfrac{0.0625}{0.1992} = 0.313$$

$$P(q_6|\mathbf{p}) = \tfrac{0.0039}{0.1992} = 0.020$$

**Figure 3.9:** Example of computation of probabilistic error correcting parsing. Here, due to space constraints, the vocabulary in Figure 3.7 has been replaced by other composed of symbols instead of words

Clearly, other distributions can be considered by applying some knowledge about the problem. For instance, we could define the editing operations at phoneme level. This way, ECP parsing would be used to find the most similar phoneme sequence given the prefix aimed at achieving a more realistic segmentation.

### 3.3.3 An adaptive learning approach to estimate the edition operations probability

The *uniform* distribution for the ECP operations proposed in section 3.3.2 is, maybe, the most natural choice when a reliable estimation of this probability can not be carried out. However, we can take advantage of the fact that in CAST, the transcriptions obtained are completely user-validated. As a consequence, we have some data about the real task to attempt a better estimation of that distribution. Next, an algorithm to estimate the editing operation probabilities in CAST will be discussed.

Firstly, we can define a set $C$ of tuples $c = (c_i, c_d, c_s, c_{wsr})$ where the first three elements in each tuple represents a point in the space of the probability distribution for the editing operations. (that is, $c_i$, $c_d$ and $c_s$, represent a possible value for the insertion, deletion and substitution probabilities respectively and $c_{wsr}$ denotes the WSR achieved by using this probability distribution on the current sentence). This set $C$ is intended to split the continuous probability space into discrete points so that the search for the best probability distribution will be performed according to these points. In order to better understand what $C$ actually represents, we can give a simple example (non ECP-related). Let's suppose that we have three possible events $a_1$, $a_2$, and $a_3$ for a specific stochastic variable $A$. From all the ways in which the probability in $A$ can be assigned to $a_1$, $a_2$ and $a_3$, we can choose some points from this space of $\Pr(A)$. For instance, the set $(0.9, 0.1, 0.0), (0.5, 0.2, 0.3), (0.3, 0.6, 0.1)$ can be defined (where the elements in each tuple represents $\Pr(A = a_1)$, $\Pr(A = a_2)$ and $\Pr(A = a_3)$ respectively). The rationale behind this is to search for the optimal distribution for $A$ only on this set of points. In our case, each point corresponds to a possible distribution for the editing operations. For instance, the point $(0.5, 0.3, 0.2)$ would denote that the insertion probability is set to 0.5 and the deletion and substitution probabilities are set to 0.3 and 0.2 respectively.

When a new fully transcribed sentence is available, we compute, for each tuple $c$ in $C$ the WSR for this sentence using the distribution defined by the first three elements in $c$. This WSR figure is accumulated in the fourth tuple element $c_{wsr}$ (that is, we store here the cumulative $WSR$ obtained for all the sentences already transcribed). Then, for the next sentence to be transcribed, we will choose, as the current ECP probability distribution, that $c$ for which the $c_{wsr}$ element is minimum.

### 3.3.4 The role of the word graph probabilities in probabilistic prefix ECP

To conclude this section, there are some issues derived from the probabilistic prefix ECP formulation and its corresponding application to word graphs that deserve a

detailed explanation.

First, we can notice that in Eq. (3.27) there is no term for the prefix acoustic probability. On the one hand, these are good news, since we can not compute this probability from the word graphs (notice that this prefix could not be in our graph). In the original CAST approach (Eq. (3.21)) we had a term for this probability. In the word graph approximation (Eq. (3.27)) this role is somehow played by $\Pr(\mathbf{x}|q_b, \mathbf{s})$. This probability has to be interpreted as the acoustic probability of the graph path reaching the state $q_b$ and then obtaining the suffix $\mathbf{s}$. The probability $Pr(q_b|\mathbf{p})$ is used, on the other hand, to find the optimal segmentation between the prefix and the suffix in the word graph.

Regarding the language model, $\Pr(\mathbf{s} \mid \mathbf{p})$ can not be properly modeled using the word graph only. When searching for a suffix after the ECP, we start from the nodes reached as result of this ECP. These nodes, however, represent paths containing *distorted* version of the prefix, as a result of the different editing operations applied. As a consequence, the language model probabilities in those nodes, might not be the proper ones. For that reason, they should be replaced by the real $n$-gram probabilities in order to benefit from the actual prefix. Figure 3.10 shows an example of this replacement.

## 3.4 Experimental results

In the following sections, our CAST experimental framework is described in detail.

### 3.4.1 Corpora

Two different tasks have been mainly used. The first one corresponds to the EU-TRANS corpus [1], composed of sentences used in conversations between a tourist and a hotel receptionist. The second one is the XEROX corpus [13], consisting of spoken utterances from printer manuals. The initial version of this corpus consisted of fragment sentence utterances aimed at testing a speech interface proposal for Computer Assisted Translation (CAT) systems discussed in Chapter 5. Then, it was later extended to be employed in CAST. The main features of both corpora are presented in Table 3.1. In addition, the well known *Wall Street Jounal* (WSJ) corpus [42] was used in the word graph CAST experiments.

Regarding the training corpora, the acoustic models, on the one hand, were estimated from the ALBAZYIN and WSJ corpora, as shown in Table 3.2. In the EU-TRANS and XEROX experiments, monophone HMMs (obtained with the HTK toolkit [55]) were employed. For WSJ, triphones were used. Speech pre-processing and feature extraction consisted in speech boundary detection, followed by the computation of the first ten MEL cepstral coefficients plus the energy, along with the corresponding first and second derivatives [30].

On the other hand, the language models for both tasks were estimated from the corpora described in Table 3.3. The SRILM toolkit [48] was used to estimate Kneser-Ney smoothed 3-grams [9].

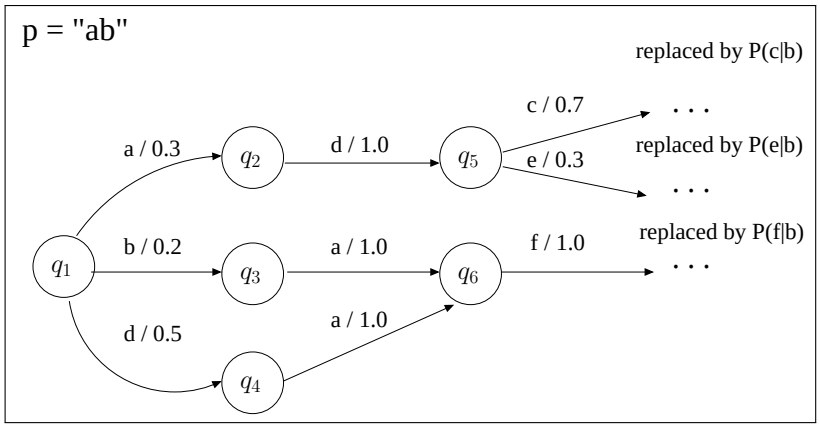



**Figure 3.10:** Example of substituion of language model probabilities after ECP. Given the previous word graph fragment, where only the language model probabilities are shown (a bigram can be assumed), let's suppose that **p**="$ab$" is the current prefix. After ECP, all the states will have an ECP probability associated and, therefore, the search for the suffix will start from any state. If we focus, for instance, on states $q_5$ and $q_6$, we can see that, in order to reach $q_5$, a substitution ($b \rightarrow d$) has to be performed. The same for $q_6$, with the substitution ($b \rightarrow a$). As a result, when starting the search for the suffix, we have the arcs leaving $q_5$ scored with the probabilities $P(c|d)$ and $P(e|d)$ where the final word in the prefix is $b$ and not $d$. The same happens with $q_6$ (in this case, with $P(f|a)$). In order to benefit from the prefix information in a better way, the probabilities in the arcs leaving $q_5$ should be replaced by $P(c|b)$ and $P(e|b)$ respectively. The same in the case of $q_6$, where the arc leaving this state should be rescored with the probability $P(f|b)$

**Table 3.1:** Features of the EUTRANS, XEROX and WSJ test corpora

| | EUTRANS | XEROX | WSJ 5K | WSJ 20K |
|---|---|---|---|---|
| Test sentences | 336 | 875 | 330 | 333 |
| Running words | 3340 | 8569 | 5683 | 5974 |
| Test-set perplexity (3-grams) | 7 | 41 | 60 | 155 |

### 3.4.2 Error Measures

The metrics used in the experiments tries to gives an estimation of the user effort required to transcribe a set of sentences through a CAST approach. Two kind of measures have been adopted. On the one hand, the *well known* word error rate (WER) has been used. On the other hand, the word stroke ratio (WSR) ([13], [12]), a measure borrowed from CAT has been employed. This measure is computed by using reference transcriptions of the speech segments considered. After a first CAST hypothesis, the longest common prefix between this hypothesis and the reference sentence is obtained, and the first unmatching word from the hypothesis is replaced by

**Table 3.2:** Features of the Spanish ALBAYZIN and English WSJ acoustic training corpus (K= $\times 1,000$)

| | Spanish ALBAYZIN | English WSJ |
|---|---|---|
| Speakers | 164 | 45 |
| Running words | 42K | 136K |

**Table 3.3:** Features of the EUTRANS, XEROX and WSJ LM-training corpora

| | EUTRANS | XEROX | WSJ 5K | WSJ 20K |
|---|---|---|---|---|
| Training sentences | 10k | 55k | 1612k | 1612k |
| Running words | 97k | 627581 | 38500k | 38500k |
| Vocabulary size | 684 | 10835 | 4989 | 19982 |

the corresponding reference word. This process is iterated until a full match with the reference sentence is achieved. The WSR is, therefore, the number of required corrections divided by the overall number of reference words.

The comparison between WER and WSR would give us an idea about the amount of effort required by a CAST user with respect to the effort needed by using a classical speech recognition system followed by a manual post-editing process (we will refer to this as *Estimated effort reduction*, EER, from now on).

### 3.4.3 Experiments

The experiments consisted in a series of block validation on the test corpora. Training is always carried out on the whole set of acoustic and text training data summarized in Tables 3.2 and 3.3. This way of proceeding slightly resembles the approach called K-fold cross validation but, in this case, one block (development) was chosen for optimizing some parameters of the search. Once these parameters have been set on the development block, the remaining blocks (test) were used as the proper test-set. This framework is aimed at trying to draw more general conclusions on the sparse test data available. In the usually employed holdout method, one single partition of the original test data into development and test sets is considered. In our experiments, the original test was split into several blocks so that different development and test sets could be derived from these blocks. Specifically, five blocks, with sizes of 67 and 175 sentences, have been considered for EUTRANS and XEROX, respectively. The experiments were actually carried out in five trials. In trial number $i$ the block number $i$ was used as development set and the four remaining blocks were used as test (here we are trying to follow a realistic approach, where only a small development set is available during the system design. The real test for the system is bigger since it consists of all the transcriptions obtained during its normal operation mode. Notice that in K-fold cross validation only one block is used as test and the remaining ones are used for training).

On the other hand, WSJ 5k and WSJ 20K corpora have been also used to test the word graph based approximation, which constitutes the most feasible technique to actually use CAST in real environments.

The development sets were specifically used to tune the *Language Model Scale Factor* which, as was mentioned, is basically a scaling factor for the second term in Eq. (3.1).

### 3.4.4 Results

In Table 3.4 the mean and the standard deviation of the results on the five test sets obtained as described in section 3.4.3 are reported. In the first two rows, a comparison between two estimations of the off-line (WER) and interactive (WSR) user effort is shown. As can be observed, significant improvements are obtained when using the CAST approach with respect to the classical ASR followed by a human post-processing approach.

In addition, the WSR results for the word graph based techniques are also presented (the results on the WSJ corpus are in Table 3.5. As can be noticed, the use of the word graphs does not affect the performance significantly, while improving the WER baseline. The results obtained by the initial ECP presented in section 3.3.1 and the probabilistic word ECP described in section (WECPP) 3.3.2 are quite similar. However, we have to take into consideration that the margin of improvement is actually constrained by the WSR results on the original CAST implementation. On the other hand, a significant number of sentences in both corpora require no interactions, as it is shown in Figure 3.11, which causes that some improvements have a small impact on the overall results. In order to clarify this a little more, the XEROX corpus has been split into different sets based on the cumulative distribution shown in Figure 3.11 (that is, the first set contains all the sentences that require at least one interaction, the second one contains the sentences that require at least two interactions and so on). Table 3.6 shows the WSR results for the baseline CAST approach, the initial ECP and the new WECPP based on this sentence distribution.

Notice that for sentences with exactly one error, the post-editing approach should be similar to CAST in effort terms, since a properly designed user interface should permit to disable the prediction engine when only a mistake is found (for sentences with more than one interaction an EER of 22.4% is achieved, as it is shown in Table 3.6).

The previous results show that the use of word graphs is competitive in terms of WSR. However,it is still necessary to check whether this new approximation can actually improve or not the system time response. To this end, the CAST system latency was measured in the following way. First, experiments corresponding to the approach described in section 3.2.3 (in the first two rows of Table 3.4) were carried out, where, for each user interaction, a complete speech recognition process is conducted. Since an exhaustive search in speech recognition is usually prohibitive, a pruned search approach was adopted in these experiments to achieve an appropriate tradeoff between accuracy and time response as Tables 3.4 and 3.7 show.

**Table 3.4:** Results obtained on the EUTRANS and XEROX corpus. The mean and the standard deviation for the test sets in the 5 block validation series are shown. The first row corresponds to the post-editing approach. The second an the third rows show the results for the interactive baseline approach and the ECP word graph based approach described in section 3.3.1 respectively. In the four row, the results correspond to the Probabilistic Word ECP discussed in section 3.3.2. Finally, the last row shows the results of the adaptive learning technique described in section 3.3.3. We can notice an EER (Estimated Effort Reduction) of about 39% and 19% for the two corpora respectively.

| | | EUTRANS | | XEROX | |
|---|---|---|---|---|---|
| | | mean | sd | mean | sd |
| Direct | WER | 7.7 | 1.3 | 22.9 | 2.4 |
| | WSR | 4.7 | 1.4 | 18.6 | 2.1 |
| Word graph | ECP WSR | 4.8 | 1.4 | 19.5 | 2.1 |
| | PWECP WSR | 4.7 | 1.3 | 19.3 | 2.1 |
| | ALPWECP WSR | 4.6 | 1.2 | 18.8 | 2.0 |

**Table 3.5:** Results obtained on the WSJ corpora. The mean and the standard deviation for the test sets in the 5 block validation series are shown. The first row corresponds to the post-editing approach. The second an the third rows show the results for ECP word graph described in section 3.3.1 AND to the Probabilistic Word ECP discussed in section 3.3.2. Finally, the last row shows the results of the adaptive learning technique described in section 3.3.3. The EER achieved is about 12%

| | WSJ 5K | | WSJ 20K | |
|---|---|---|---|---|
| | mean | sd | mean | sd |
| WER | 6.2 | 1.5 | 10.6 | 1.7 |
| ECP WSR | 5.9 | 1.3 | 9.9 | 1.6 |
| PWECP WSR | 5.6 | 1.4 | 9.5 | 2.0 |
| ALPWECP WSR | 5.5 | 1.3 | 9.3 | 2.1 |

**Table 3.6:** Results (WER and WSR) on XEROX corpus for different CAST techniques based on the distribution of the sentences shown in Figure 3.11. The baseline column shows the results obtained by the original CAST approach without using word graphs

| | WER | WSR | | |
|---|---|---|---|---|
| | | Baseline | ECP | WECPP |
| 1 interaction or more | 39.6 | 33.1 | 35.1 | 34.3 |
| 2 interactions or more | 50.9 | 40.1 | 44.3 | 43.2 |
| 3 interactions or more | 54.6 | 45.6 | 51.0 | 49.8 |

In the case of the word graph approaches, we have to take into account two different kinds of computations. Firstly, we have to generate the word graph from the input signal. This process entails a standard speech decoding plus some additional work necessary to obtain the word graph. Nevertheless, it is reasonable to assume that we can generate the word graphs "in advance" before starting a CAST session (or as a look-ahead background computation). This assumption is based on the fact that, in our case, speech transcription is carried out from recorded signals. Therefore, we can consider the construction of the word graphs as a batch and separate process from the interactive transcription task itself. In any case, this word graph generation time is included in the third row in Table 3.7 for informative purposes. In the case of the direct approach, it is not possible to perform any off-line work apart from the usual signal pre-processing and feature extraction. To summarize, the interactive word graph time response is exclusively given by the cost of the search for the suffix on the word graphs.

As expected, the *word graph* approach notably outperforms the baseline. Especially, in the case of XEROX, where the baseline technique seems to be too slow to be even considered and the word graph approach proves to be the best solution to implement CAST in a real environment

In addition to this, we can expect a diminishing time response when using word graphs as the number of interactions grows for a sentence. Whereas the initial system hypothesis is actually a whole sentence prediction, following predictions tend to be shorter as the prefix length increases. Since the computational cost of the prefix parsing is significantly lower than the search for the suffix, the time response goes down. To quantify this fact, Figure 3.12 shows the average response (XEROX CORPUS) time based on the specific number of interaction performed (that is, the first point in the graph is for the initial system prediction, the second one for the prediction after one user interaction and so on). The cumulative distribution histogram shown in Figure 3.11 for the XEROX may help to better understand the previous results.

In order to give a reference point for the different time results, we can mention that all the experiments were performed on a 3.2 Ghz Intel Xeon CPU.

**Table 3.7:** Average interaction time response. The first row shows the time response of the baseline approach to CAST. The second row reports the interactive time response of the word graph approximation. Finally, in the third row the average time needed to generate the word graphs is shown. All the times are in seconds

| Approach | EUTRANS | XEROX |
|---|---|---|
| Baseline CAST | 0.9 | 3.3 |
| word graph CAST | 0.4 | 0.5 |
| Including word graph generation time | 1.7 | 1.9 |

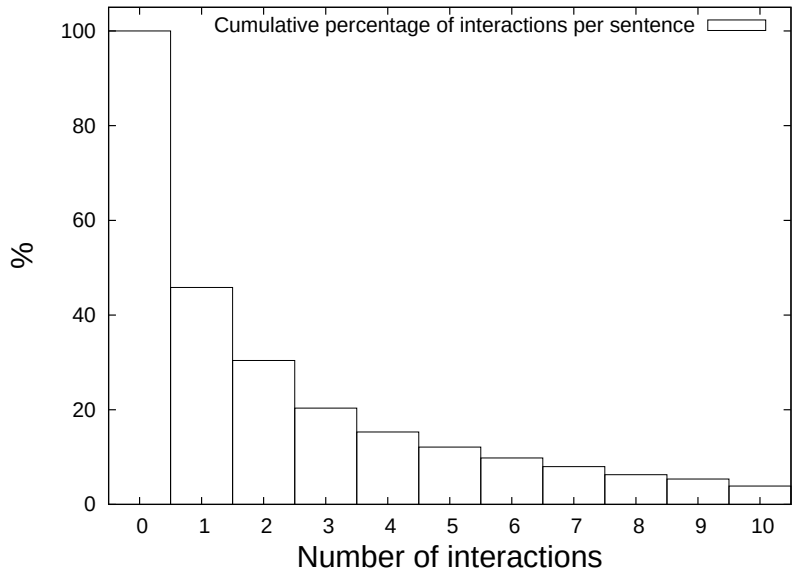



**Figure 3.11:** Cumulative sentence distribution based on the number of user interactions needed to obtain a perfect transcription. The first bar shows the percentage of sentences that are perfectly transcribed with zero o more interactions (the whole corpus in this case), the second bar the percentage for one or more interactions and so on

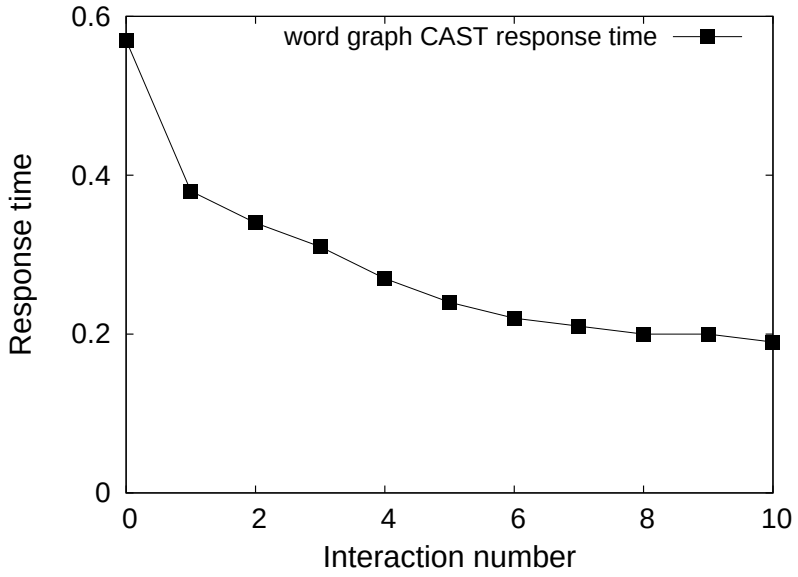



**Figure 3.12:** Average word graph CAST time response based on specific interaction number within a sentence

## 3.5 Summary of contributions

A new approach to the production of perfect speech transcriptions has been presented. This approach combines the efficiency of an automatic speech recognition system with the accuracy of a human transcriber. Firstly, a direct implementation of this system has been described and some experiments have been carried out to assess the improvements that these techniques can achieve.

Next, an alternative proposal aimed at improving the system efficiency has been presented. In the first place the use of word graphs along with error correcting parsing has been tested to, posteriorly, propose a formulation to properly integrate both techniques. Finally, an adaptive learning method has been proposed aimed at learning the ECP probabilities for each specific task by using the information obtained from the user feedback.

CHAPTER 4

# INTERACTIVE TEXT GENERATION

## 4.1 Introduction

Since the adoption of the written language by the ancient human societies, writing texts has become a very common task. The discovering of electronic computers has significantly facilitated this process. Computers allow us to generate text faster and more comfortably than before. However, so far, the approach adopted when using a computer is basically the same as centuries ago. Computers are, essentially, much more sophisticated replacements of paper, pencil and eraser, but typing text is still a manual process and the incredible computational power that computers provide are barely used here. Only tools like orthographic and grammatical checkers along with thesaurus are generally employed.

Developing automatic assistance system in this field can be completely worth it since the time spent in typing (or in thinking about what it is going to be typed) is quite high in many environments. A system able to predict (with, of course, some degree of accuracy) what someone is going to type and thereby saving considerable amounts of effort could be utterly helpful.

In some situations, on the other hand, typing becomes a too slow and uncomfortable task. In some devices, such as mobile phones, no suitable input interfaces have been developed. Besides, some disabled people are not able to achieve a sufficient typing speed and, unfortunately, this can be the only way for them to communicate.

Different approaches to this topic can be found in the literature. Most of them only attempt to predict the next single word [50] and/or they are concerned about measuring the accuracy of off-line text predictions [2]. Here, we consider a more general setting where not only single words but multi-word fragments or full sentences are predicted under an interactive paradigm.

### 4.1.1 Interactive Text Generation and Interactive Pattern Recognition

Providing assistance in text typing is a task that can be easily included in our Interactive Pattern Recognition approach. The basic process would consists in predicting (completing) some portion of text based on the text previously typed. Using the terminology adopted in this approach, we would have to find a suitable continuation (suffix) for a given prefix. However there is a big difference in this case that should be discussed before going further.

In the general IPR framework, the goal is to decode some input signal or data. In our proposal for text prediction, no input is available and the system just has to find the most suitable text according to the prefix. In other words, the user feedback is the only thing we can use to produce an outcome (actually, this formulation can be interpreted in a different way by considering the prefix as the input pattern leading to a *classical* Pattern Recognition problem). This fact makes this task much harder since the system hypotheses can not be derived and constrained by an input pattern. For that reason, the possible set of suitable system outputs is much bigger and, therefore, we can expect a considerable drop in the system accuracy.

We can formalize this process as the search for the most likely continuation (suffix) $\mathbf{s}$ given the text typed so far (prefix) $\mathbf{p}$,

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\operatorname{argmax}} \Pr(\mathbf{s}|\mathbf{p}) \tag{4.1}$$

This is similar to what was described for CAST. Initially, no prefix (feedback) is available, and the system makes an initial prediction. The user validates the prediction, selects an error-free prefix and adds some text to this prefix. Then, the system will complete this user-validated and corrected text until the whole prediction is accepted.

It could be interesting to discuss some practical aspects that arise in this new task. On the one hand, the initial prediction (when no prefix is available) will be always identical since the conditional probability in Eq. (4.1) only depends on the prefix and it is empty. Because of this, an initial prediction can be useless (or, in other words, very inaccurate) and a better approximation is to wait for the user to type something before starting predicting. Nonetheless, there can be some scenarios where a initial prediction could be justified (for instance, documents starting with a fixed sequence of words). This can be merely regarded as an implementation issue and the use of prediction models for this case is not even worth mentioning.

In addition to this, in CAST, we have an input that somehow help us to determine the prediction length. On the contrary, in text generation, a new strategy has to be developed in order to know when to stop generating words. Predicting just one word after the prefix seems to be the easiest thing to do but we could reach more benefit by predicting multi-word fragments.

Predicting whole sentences, on the other hand, could be a good choice but a full-sentence language model is needed and this kind of models has not been proved to be actually useful in language modelling. Letting the user set the length of predictions

is, maybe, a good alternative but letting the system itself deal with this problem could be worth it. This issue will also be discussed later.

## 4.2 Developing an interactive text generation system

Once the problem has been introduced, we are going to address the design and implementation of an Interactive Text Generation (ITG). We will begin by describing the models involved in ITG.

### 4.2.1 Language modelling. Using $n$-grams

As was discussed in the previous chapter, $n$-grams [21] are the most widely used language models in NLP applications. Our approach to text generation also lies on $n$-grams but taking into account some considerations that are discussed in the following paragraphs. Basically, the idea when adopting this kind of models is to take advantage of the $n-1$ last words in the prefix to suggest an appropriate continuation. Clearly, these models fail in benefiting from the whole information available and just a small portion of it is actually considered.

As was described in section 4.1.1 the process consists in, given a user prefix $\mathbf{p} = \mathbf{w}_1^k$, finding the optimal completion $\hat{\mathbf{s}}$ to this prefix. Let $\mathbf{s} = \mathbf{w}_{k+1}^l$ be a possible suffix hypothesis of arbitrary length $l-k$. The language model used here is identical to the one specified for CAST. Nevertheless we can write, as a remainder:

$$\begin{aligned} \Pr(\mathbf{s} \mid \mathbf{p}) &\approx \prod_{i=k+1}^{k+n-1} P(\mathbf{w}_i \mid \mathbf{w}_{i-n+1}^{i-1}) \cdot \prod_{i=k+n}^{l} P(\mathbf{w}_i \mid \mathbf{w}_{i-n+1}^{i-1}) \\ &= \prod_{j=1}^{n-1} P(\mathbf{s}_j | \mathbf{p}_{k-n+1}^{k}, \mathbf{s}_1^{j-1}) \cdot \prod_{j=n}^{l-k} P(\mathbf{s}_j | \mathbf{s}_{j-n+1}^{j-1}) \end{aligned} \tag{4.2}$$

### 4.2.2 Searching for a suffix

Next, we are going to focus on the search problem for ITG. By strictly following Eq. (4.1), where a conditional probability has to be maximized, we could apply the usual Dynamic Programming based approach [54]. Notice however that, because only a language model is used, the decoding process is easier than in CAST and only consists in constructing suffixes according to this LM. At this point, we can anticipate a problem that will be discussed later. In a usual Pattern Recognition task, we have an input pattern to decode and the algorithm can be stopped when the end of the pattern is reached. In this task, we do not have an input and, therefore, we do not have any clue to stop predicting words. Moreover, we have to cope with another problem related to the nature of the $n$-gram model since longer predictions can be penalized over shorter ones. For the time being, we will define a generic function, $F_{length}(.)$,

which, given a prediction hypothesis, returns a score for this hypothesis according to its length.

Surprisingly enough, the maximization of the posterior probability and the entailed *Viterbi* search is not necesarily the best search strategy in this case. Recently, a better and simpler approach has been proposed [40].

### 4.2.3 A greedy algorithm to predict suffixes

We have just claimed that the usual search strategy in Pattern Recognition is not the best thing we can do here. We will call this strategy *MaxPost* from now on. Let's see why it is not optimal in this case.

*MaxPost* is actually aimed at minimizing the decoding errors. In NLP, this criterion optimizes the number of sentences correctly predicted (here, we are considering that each input pattern is decoded as a sentence [a]). In other words, the well-known Sentence Error Rate (SER) metric is the optimization goal for *MaxPost*.

In an interactive task like ITG, the real goal is to save user interactions and not necessarily maximizing the number of correct sentences for the general case. According to this, in [40] an optimal strategy to predict suffixes in an interactive environment is achieved. This strategy turns out to be a greedy-like search (denoted as *Greedy* from now on) and it is based on constructing the final hypothesis by taking just optimum local decisions.

The superiority of the *Greedy* approach could be alternatively derived by applying the optimal classification rule properly. Let's see how. If we consider a scenario where exactly one word is predicted after the prefix, only two possibilities arisen. If the prediction was correct, the word is added to the prefix (thereby generating a new prefix) and the process is iterated again. If the word was not correctly predicted, a word stroke is computed and the correct word is added, again, to form a new prefix. It is easy to see that this scenario is completely equivalent to the more general one with respect to the number of word strokes needed to produce a sentence. Therefore, the conclusions reached here can be applied, as well, to our multi-word prediction case.

By taking this new point of view, we can consider that we are addressing an iterative classification problem where, in each iteration, we have a prefix and we obtain a label class (the word predicted) for this prefix. By making the reasonable assumption that each classification step is independent from the previous one, we can directly apply the optimal decision rule. This rule tells us that we have to maximize the posterior probability of the class (word) given the pattern (prefix). This way, we should choose, in each iteration, the most probable word given the prefix, which turns out to be, indeed, a greedy prediction algorithm.

Anyway, it is interesting to deeply analyze the behavior of both *MaxPost* and *Greedy* in the interactive scenario. To this end, we are going to rely on the simple model shown in Figure 4.1.

[a] In most cases, this approach is followed since working with too long inputs is not practical and, hence, the input is previously segmented into sentences

**Algorithm 2**: Viterbi-based algorithm to search for the best continuation to a prefix. $n$-gram states are identified as substrings of length $n-1$. Therefore, if (for example) $n=3$, $q=\mathbf{w}_{i-n+2}^{i}$ denotes a state identified as the 2-gram $\mathbf{w}_{i-1}\mathbf{w}_i$. It is assumed that, if $j<i$, $\mathbf{w}_i^j$ is the empty string ($\lambda$). The function $F_{length}(i,g)$ provides the length-conditioned score for an $i$-words sentence with likelihood $g$. Different implementations of this function are discussed in section 4.2.4.

**input** : user validated prefix ($\mathbf{p}$), vocabulary ($V$), maximum prediction length ($maxLen$), $n$-gram size ($n$), length score function ($F_{length}$)
**output**: whole sentence prediction
**begin**
  $i=|\mathbf{p}|+1$; $q=\mathbf{p}_{i-n+1}^{i-1}$;
  $Q=\{q\}$; // States
  $G[q]=0$; // Likelihoods;
  $W[q]=p$; // Word sequences
  $g_{best}=0$; $w_{best}=\lambda$
  **while** $i<maxLen$ **do**
    $Q'=Q$; $G'=G$;
    $W'=W$; $Q=\emptyset$;
    **forall** $q'\in Q'$ **do**
      **forall** $v\in V$ **do**
        $q={q'}_2^{n-1}\cdot v$ // concatenate $v$ to $\mathbf{w}_{i-n+2}^{i-1}$
        **if** $q\notin Q$ **then**
          $Q=Q\cup\{q\}$;
          $G[q]=G'[q]\,P(v|q')$;
          $W[q]=W'[q']\cdot v$;
        **else if** $G[q]<G'[q]\,P(v|q')$ **then**
          $G[q]=G'[q]\,P(v|q')$;
          $W[q]=W'[q']\cdot v$;
    $g^*=0$; $\mathbf{w}^*=\lambda$;
    **forall** $q\in Q$ **do**
      **if** $G[q]>g^*$ **then**
        $g^*=G[q]$;
        $\mathbf{w}^*=W[q]$; //best result for length $i$
    **if** $g_{best}<F_{length}(i,g^*)$ **then**
      $g_{best}=F_{length}(i,g^*)$;
      $\mathbf{w}_{best}=\mathbf{w}^*$; // best result so far
    $i=i+1$;
  **return** $\mathbf{w}_{best}$;
**end**

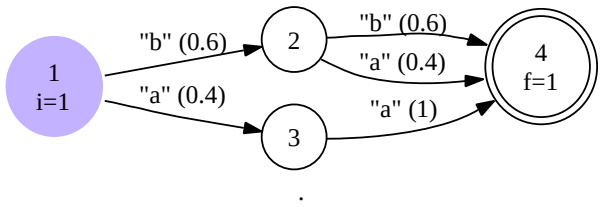



**Figure 4.1:** Simple stochastic language model

Let's suppose that we generate the set of strings modeled by this example. This set is composed of the strings *aa*, *ba* and *bb*. Now, we can compute the expected number of interactions ($I$) required to generate these strings when using *MaxPost*:

$$E_{MaxPost}(I) = 0 \cdot 0.4 + 1 \cdot 0.24 + 2 \cdot 0.36 = 0.96 \tag{4.3}$$

That is, the string $aa$ is generated with probability of 0.4 and does not need any interaction (notice that $aa$ represents the most probable path in the automaton). In the case of the string $ba$, it is generated with probability of 0.24 and it needs one interaction (in the first one, with no prefix, $MaxPost$ predicts the string $aa$ and after setting $b$ as a prefix, the string $ba$ is finally achieved). Finally, the string $ba$ is generated with probability of 0.36 and two interactions are needed in this case.
In the case of *Greedy*, this expected value is given by:

$$E_{Greedy}(I) = 1 \cdot 0.4 + 1 \cdot 0.24 + 0 \cdot 0.34 = 0.64 \tag{4.4}$$

Here, *Greedy* starts by predicting $bb$ when no prefix is available (as the third term in the equation above shows) and two and one interactions are needed for the strings $aa$ and $ba$ respectively (as the first and second term indicates).

Therefore, the *Greedy* approach is expected to require less overall interaction effort. But what can be expected about the prediction errors?. We can notice that the (expected) number of prediction errors corresponds to the (expected) number of interactions. When the system fails to generate a perfect suffix a user interaction is required. Therefore, we can conclude that, in this case, the number of prediction errors is minimized by *Greedy* and not by *MaxPost*. proved to be optimal.

From the above example, an apparent paradox about the optimality of the *MaxPost* approach is reached. The previous reasoning does not try to disprove the well known optimal classification rule. This *contradiction* is solved when properly considering the real context of this rule. We can, in the previous example, compute the expected full-length prediction errors for the general case (not interactivity here) considering all the possible situations. Specifically, we have three different classification problems given by the three different prefixes that the strings in the model can have.

These prefixes are $\lambda$, $a$ and $b$. In this simple model, when generating all the possible strings, these prefixes occur with probabilities $1$, $0.4$ and $0.6$ respectively. Thus, the expected value for *MaxPost* in the general case can be expressed as:

$$E(C)_{MaxPost} = 1 \cdot 0.6 + 0.4 \cdot 0 + 0.6 \cdot 0.4 = 0.84 \tag{4.5}$$

while for *Greedy*, we have:

$$E(C)_{Greedy} = 1 \cdot 0.64 + 0.4 \cdot 0 + 0.6 \cdot 0.4 = 0.88 \tag{4.6}$$

This result actually shows that the interactive scenario is quite different and the conclusions reached for the general case do not directly apply in this kind of tasks.

Now, we only have to formalize this greedy approach for the case of $n$-grams based ITG. This is described in detail in Algorithm 3.

**Algorithm 3**: Greedy strategy to complete a user-validated prefix. Note that the greedy solutions shorter than *maxLen* are just the prefixes of the resulting $w$.

**input** : user validated prefix ($\mathbf{p}$), vocabulary ($V$), maximum prediction length ($maxLen$), $n$-gram size ($n$)
**output**: whole sentence prediction ($\mathbf{w}$)
**begin**
  $\mathbf{w} = \mathbf{p}$; $i = |\mathbf{p}| + 1$;
  **while** $i < maxLen$ **do**
    $v^* = \lambda$; $g^* = 0$;
    **forall** $v \in V$ **do**
      **if** $g^* < P(v|\mathbf{w}_{i-n+1}^{i-1})$ **then**
        $g^* = P(v|\mathbf{w}_{i-n+1}^{i-1})$;
        $v^* = v$;
    $\mathbf{w} = \mathbf{w} \cdot v^*$;
    $i = i + 1$;
  **return** $\mathbf{w}$;
**end**

### 4.2.4 Dealing with sentence length

We have already introduced one of the problems arisen in ITG related to obtain a suitable strategy to stop generating words. This problem, as was mentioned, could be regarded (for simplicity) as one of the practical issues of the system. This way, simple solutions such as letting the user set a maximum prediction length could be adopted. Nevertheless, there is also an important problem related to the nature of the models used in predictions that can not be overlooked. When using a dynamic programming approach, a trellis containing all the explored hypotheses is constructed. Each stage of the trellis contains same-length hypotheses. Initially, the 1-word hypotheses are

considered. Next, 2-word hypotheses are generated and evaluated and so on. On the other hand, an $n$-gram language model scores a sentence by computing the product of the probabilities of all the $n$-grams present in the sentence (see Eq (4.2) for the ITG case). Since all these probabilities are numbers between 0 and 1, we can say that, in general, the fewer amount of $n$-grams in the hypothesis the higher the score is or, in other words, shorter predictions would have (in average) a better score than longer ones (in the most extreme case, our system would always produce a 1-word word prediction).

We are going to propose two different alternatives to approach this problem. The first one is based on normalizing each hypothesis by its length. This normalization can be better expressed by following the usual log-prob computation:

$$score(\mathbf{s}|\mathbf{p}) = \frac{\sum_{i=k+1}^{l} logP(\mathbf{w}_i \mid \mathbf{w}_{i-n+1}^{i-1})}{l} \tag{4.7}$$

According to this formula, the best hypotheses are those whose individual $n$-gram probabilities are higher on average. This approach, however, presents an important drawback that should be taken into account: because of the normalization, this model is not a probabilistic model anymore and some of the desirable properties that characterizes this kind of models are now missing.

We propose, in addition, a different approach. We can rely on a separate model to account for the length (denoted as $P_{len}$). For instance, a Gaussian can be chosen as a distribution over all the possible lengths. This Gaussian distribution is trained by a maximum likelihood criterion on the training samples. Once a explicit length model is available, a linear interpolation is performed between the $n$-gram model and this new length model as it is shown in Eq. (4.8).

$$P(\mathbf{s}|\mathbf{p}) = \alpha \cdot P_{len}(|\mathbf{p}| + |\mathbf{s}|) + (1-\alpha) \cdot \prod_{i=k+1}^{l} P(\mathbf{w}_i \mid \mathbf{w}_{i-n+1}^{i-1}) \tag{4.8}$$

In the case of the *Greedy* approach, this is not actually an issue. In *MaxPost*, multiple partial hypotheses are considered in parallel. In *Greedy*, however, a single prediction is constructed by inserting words at the end of the hypotheses. That means that the length of the prediction does not significantly modify the content of the prediction itself (but only the number of words). In other words, the best prediction of length $m$ will be identical to the best prediction of length $m+1$ except that, in this last case, we have an additional final word.

## 4.3 Experiments

The evaluation method proposed in section 2.6 will be followed here. We will use, for the time being. the already defined Word Stroke Ratio (WSR). In Figure 4.2 an example of an ITG session and the corresponding WSR computation is shown.

| | |
|---|---|
| **Iteration 1** | |
| *Prediction:* | `Check the printer before sending jobs` |
| *Prefix:* | **Check the** |
| *Amendment:* | **Check the** *following* |
| **Iteration 2** | |
| *Prediction:* | **Check the following** `conditions before continuing` |
| *Prefix:* | **Check the following conditions** |
| *Amendment:* | **Check the following conditions** *to* |
| **Iteration 3** | |
| *Prediction:* | **Check the following conditions to** `ensure an optimum work` |
| *Prefix:* | **Check the following conditions to ensure an optimum** |
| *Amendment:* | **Check the following conditions to ensure an optimum** *performance.* |

RESULT: **Check the** *following* **conditions** *to* **ensure an optimum** *performance.*

$WSR = \frac{3}{9} = 0.33 \rightarrow 33\%$

**Figure 4.2:** Example of editing session and the corresponding WSR computation. The system generates an initial prediction. Then, the user validates a correct prefix (boldfaced) and introduces a word amendment (shown in italics). The system, taking into account this information, generates a new prediction. The process is iterated until a correct, full sentence is achieved. In the final result, the user only had to type the two words shown in italics. The WSR is obtained by dividing the number of user word strokes between the overall number of words.

### 4.3.1 Corpora

Three different tasks are considered. The first two were already used in CAST, EUTRANS and XEROX. The third one, ALBAYZIN, consists on a set of natural language based queries to a geographic database. In Table 4.1 the main features of these corpora are shown.

### 4.3.2 Results

The experiments performed in this section are aimed at evaluating two different things. On the one hand, the accuracy of the ITG proposal has to be measured. On the other, a comparison between the two techniques *MaxPost* and *Greedy* is needed to validate the optimality of the last one. In Table 4.2 the main results of the experiments are shown. In this table, the *Length Model* column refers to the linear interpolation showed in Eq. (4.8). The final column corresponds to apply a length normalization Eq. (4.7) in the case of the Viterbi algorithm, and to generate predictions of a defined length in the case of the *Greedy*. The best result for each corpus is shown in boldface.

We can see that *Greedy* significantly outperforms *MaxPost* in all the tasks. It is

**Table 4.1:** Features of the corpora used

| | EUTRANS | ALBAYZIN | XEROX |
|---|---|---|---|
| Test sentences | 2996 | 1440 | 875 |
| Running words | 35023 | 13566 | 8257 |
| Running characters | 188707 | 81246 | 53337 |
| Training vocabulary | 688 | 1271 | 10913 |
| Training sentences | 10000 | 9893 | 53740 |
| Test-set perplexity (3-grams) | 4.9 | 6.6 | 41 |

**Table 4.2:** WSR results on different corpora. A comparison between the two search algorithm proposed is shown for the three corpus considered. The columns under *Length model* show the result of interpolating the $n$-gram with a probabilistic length model under different values for the $\alpha$ parameter in Eq. (4.8). The column under *Length normalization* shows the result of applying a length normalization on this algorithm. The final column reports the results achieved by the greedy approach.

| | *MaxPost* | | | | | | *Greedy* |
|---|---|---|---|---|---|---|---|
| | Length model ($\alpha$) | | | | | Length norm. | |
| Corpus | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 | | |
| EUTRANS | 57.6 | 57.6 | 60.4 | 62.7 | 62.7 | 62.5 | **50.9** |
| ALBAYZIN | 62.5 | 62.5 | 62.5 | 62.5 | 62.8 | 60.4 | **53.6** |
| XEROX | 79.6 | 79.6 | 79.7 | 80.0 | 80.0 | 77.3 | **66.3** |

noticeable, as well, that in simple tasks the system can accurately predict about half of the overall words in the reference sets. It is also worth mentioning that the *Xerox* test is the same used in CAST. By comparing both results (a WSR of 18.6 was achieved in CAST) we can get a picture about the difference in accuracy between a strict IPR application. Additionally, the full-length prediction errors achieved by both methods (see section 4.2.3) are reported in Table 4.3.

**Table 4.3:** Prediction (classification) errors on different corpora. A classification is considered correct when a full error-free suffix is predicted. Notice that the number of classification performed is different for each algorithm since this figure corresponds to the number of interactions needed to generate the reference set.

| | Corpus | | |
|---|---|---|---|
| Approach | EUTRANS | ALBAYZIN | XEROX |
| *MaxPost* | 19973 (95.8%) | 8754 (96.1 %) | 6567 (97.8 %) |
| *Greedy* | 17844 (93.0%) | 7269 (95.4 %) | 5485 (94.7 %) |

Once that an initial ITG proposal has been addressed and several experiments have been reported, we are going to devote the rest of the chapter to study different extensions to this ITG core.

## 4.4 Predicting at character-level

The results shown in the previous section were obtained by considering each user interaction as a word correction (WSR). However, an alternative arises if we consider a character-based approach, that is, a system able to respond to single keystrokes rather than whole words. It is, in principle, clear whether the typing effort should be measured in terms of word or key strokes. For constrained interfaces, where the bottleneck is given by the typing mechanism, computing keystrokes seems to be reasonable, since a significant amount of time is spent in introducing the information instead of thinking about what it is going to be typed.

In this case, as soon as the user types a single character, the system provides a continuation without waiting for a full word correction. The process is essentially the same as described for our ITG initial approach but, taking into account that, when searching for the suffix, we have to deal with incomplete words (that is, the final characters of the suffix can be a word-prefix and not necessarily a whole word). Under this premise, we firstly have to complete the final characters in the prefix (that will be, usually, an incomplete word). Formally, let $c_{\mathbf{w}_k}$ be the sequence of *characters* that comes after the last blank in the prefix. We have to search for a word $\hat{v}$ for which $c_{\mathbf{w}_k}$ is a prefix. In the case of an $n$-gram language model, this amounts to the following optimization equation:

$$\hat{v} = \underset{v \in V : c_{\mathbf{w}_k} \in pref(v)}{\text{argmax}} P(v|\mathbf{w}_{k-n+1}^{k-1}) \tag{4.9}$$

where $pref(v)$ denotes the set of all the prefixes of the word $v$.

Now we have a way to deal with incomplete words in the prefix and, therefore, we know how to construct a system able to react to single key strokes. The only thing we need is a new way to measure the system performance in this prediction modality. We can accomplish this goal by means of an adequate extension of the WSR metric. This extension is called Key Stroke Ratio (KSR) and it is defined as the number of key strokes needed to achieve the reference tesx divided by running characters in this text. In Figure 4.3 an example of interaction with a character-ITG system is shown along with an example of KSR computation.

In Table 4.4, the results of this new interaction modality on the corpora described in section 4.3.1 are shown.

At this point, it can be interesting to thoroughly discuss the measure (WSR or KSR) to adopt to better estimate the effort reduction achieved. In a normal situation where a user generates a whole text document in a desktop computer, it is not clear which estimation is more reliable. If the system is planned to help a user about how to write a document (that is, to suggest grammatical constructions, specific words,

**Iteration 1**
*Prediction:* `Check the printer before sending jobs`
*Prefix:* **Check the**
*Amendment:* **Check the** *f*
**Iteration 2**
*Prediction:* **Check the f**`inal configuration before continuing`
*Prefix:* **Check the f**
*Amendment:* **Check the f***o*
**Iteration 3**
*Prediction:* **Check the fo**`llowing conditions to ensure an optimum work`
*Prefix:* **Check the following conditions to ensure an optimum**
*Amendment:* **Check the following conditions to ensure an optimum** *p*
**Iteration 4**
*Prediction:* **Check the following conditions to ensure an optimum p**`erformance .`

RESULT: **Check the *fo*llowing conditions to ensure an optimum *p*erformance .**

$$KSR = \frac{4}{65} = 0.06 \rightarrow 6\%$$

**Figure 4.3:** Example of an editing session and the corresponding KSR computation. The system generates an initial prediction. Then, the user validates a correct prefix (boldfaced) and introduces an amendment (shown in italics). The system, taking into account this information, generates a new prediction. The process is iterated until a correct, full sentence is achieved. In the final result, the user only had to type the three characters shown in italics. An final acceptation keystroke is also assumed. The KSR measure is obtained by dividing the number of user strokes between the overall number of characters.

**Table 4.4:** Results of predicting at character level . These KSR results correspond to the *Greedy* approach. The WSR results are also added for informative purposes

| Corpus | KSR | WSR |
|---|---|---|
| EUTRANS | 14.1 | 50.9 |
| ALBAYZIN | 13.3 | 53.6 |
| XEROX | 19.5 | 66.3 |

etc.), WSR seems quite adequate since words can be considered as the minimum meaningful units in human language and, therefore, the effort should be expressed in these terms.

On the other hand, if ITG is to be used as an assistance tool in problematic envi-

**Table 4.5:** Features of the SHAKESPEARE and EUROPARL corpora

| | | SHAKESPEARE | EUROPARL |
|---|---|---|---|
| Training Set | sentences | 4377 | 133772 |
| | running words | 103937 | 3884947 |
| | running characters | 481088 | 22414616 |
| | vocabulary size | 8462 | 60688 |
| Test Set | sentences | 1211 | 2195 |
| | running words | 26134 | 62017 |
| | perplexity | 211 (3-grams) | 109.1 (3-grams) |

**Table 4.6:** Results at word and character level . These results correspond with the greedy approach and predicting sentences of defined length. The prediction length was set as the mean plus the variance of the sentence-length distribution in the training set.

| Corpus | WSR | KSR |
|---|---|---|
| SHAKESPEARE | 86.9 | 43.6 |
| EUROPARL | 77.4 | 28.0 |

ronments, where the effort needed to merely typing is significant, then KSR is clearly the metric to be adopted. The results obtained so far (on *easy* tasks) seem to suggest that, for the time being, ITG is only of moderate help to solve the first situation. On the contrary, the KSR results indicate that ITG turns out to be an interesting tool in a constrained-typing situation. To corroborate this fact, additional experiments on new (and considerably) more difficult and realistic tasks will be conducted.

First, we are going to use part of the EUROPARL corpus [25], which is composed of different transcriptions of the European Parliament sessions. The second task is based on a corpus obtained from William Shakespeare plays (four plays were used as training material and one play was used as test set). In Table 4.5 the features of both corpora are shown. The results obtained (both WSR and KSR) are presented in Table 4.6.

We can see how the WSR figures are far from being satisfactory. On the contrary, promising results are achieved when trying to save key stroke effort even in these hard tasks. Next, we are going to discuss additional proposals aimed at improving the KSR results achieved in this section.

### 4.4.1 Predicting at character level using character LMs

Very often, in NLP applications, word-based language models are used for several reasons. On the one hand, words are the minimal syntactic and semantic units in the human language. On the other hand, the use of word-based models (rather than sub-words units) prevent the system from generating incorrect lexical components in the final result.

**Table 4.7:** KSR Results on the different corpora using character LMs for different $n$-gram order. The missing results in the table are due to the excessive amount of memory required to train the models. The best result for each corpus is boldfaced. In the final row, as a recordatory, the word-based lm results are reported.

| order | EUTRANS | ALBAYZIN | XEROX | SHAKESPEARE | EUROPARL |
|---|---|---|---|---|---|
| 5 | 18.9 | 20.4 | 30.1 | 44.7 | 38.9 |
| 6 | 15.8 | 17.5 | 27.0 | 43.5 | 34.5 |
| 7 | 14.3 | 15.8 | 25.1 | **43.5** | 32.1 |
| 8 | 13.9 | 14.8 | 23.3 | 43.6 | 30.7 |
| 9 | **12.9** | 14.0 | 22.0 | 44.1 | **29.6** |
| 10 | 13.6 | 12.4 | 20.8 | 50.6 | 30.5 |
| 11 | 13.5 | 11.9 | 19.9 | 52.6 | 31.6 |
| 12 | 13.5 | 11.1 | 19.5 | 54.0 | – |
| 13 | 13.8 | 10.7 | 19.3 | 54.7 | – |
| 14 | 14.1 | **10.5** | **19.2** | 55.0 | – |
| 15 | 14.4 | 10.8 | 19.3 | 55.2 | – |
| word LM | 13.1 | 13.3 | 19.5 | 43.6 | 27.7 |

In the case of character-ITG, the use of units apart from words can be more justified since the system has to deal with prefixes that can contain incomplete words. In the approach described before, the word completion was carried out by selecting all the words in the vocabulary which are "compatible" with the final characters in the prefix and then choosing the one maximizing the probability of a word-based language model. This, however, poses some problems. For instance, the lack of suitable words in the system vocabulary to perform this completion.

As an alternative, a character-based language model could be employed to directly deal with this situation. Besides, the *Greedy* algorithm can be properly applied by using this kind of model in a character-level prediction (notice that the formula in Eq. 4.9 is not exactly a greedy approach when considering individual characters). The actual algorithm for predicting the suffix in this situation is quite similar to the one in Figure 3 and, for that reason, it is omited here.

Regarding the $n$-gram model, it is clear that the order of this *new* model should be (considerably) higher than the previous word models. Instead of using 3-gram or 4-gram we will consider $n$-grams with n between 5 and 15. In Table 4.7 the results of the experiments conducted with these character models are shown. In general, the use of character language models outperfoms the previous approach although this improvement is quite slight in most cases.

### 4.4.2 Predicting at character level using both word and character LMs

Since the character language models have proven to be competitive for ITG, it seems reasonable to discuss a possible use of both kind of models to improve the prediction quality. The most straightforward approach is, perhaps, to perform a linear interpolation between the two models so that both character and word histories are considered for each hypothesis. Nevertheless, the word models are not able to directly deal with hypotheses that consist of incomplete words (in this case, these model will not be used to complete a partial word hypothesis, as in Eq. (4.9) but rather to score a partial word hypothesis). On account of this, it is mandatory to somehow allow these models to cope with this situation. A possible solution consists in scoring a the word-prefixes ($c_{\mathbf{w}_k}$) as:

$$P(c_v|\mathbf{w}_{k-n+1}^{k-1}) = \sum_{c_{\mathbf{w}_k} \in\, pref(v)} P(v|\mathbf{w}_{k-n+1}^{k-1}) \tag{4.10}$$

where $pref(v)$ denotes the set of all the prefixes (from length 1 to $|v|$) of the word $v$.

This way, all the possible whole-word alternatives are considered for the current incomplete word.

**Table 4.8:** Results (KSR) of predicting with character and word models . The last column show the best results achieved so far for each corpus

| Corpus | Interp weight | | | |
|---|---|---|---|---|
| | 0.1 | 0.3 | 0.5 | base |
| EUTRANS | **12.8** | 13.1 | 15.7 | 12.9 |
| ALBAYZIN | **10.5** | 10.9 | 14.6 | 10.5 |
| XEROX | **18.0** | 18.3 | 19.8 | 19.5 |
| SHAKESPEARE | 43.1 | 42.5 | **42.3** | 43.6 |
| EUROPARL | 28.2 | **27.7** | 27.9 | 28.0 |

As can be seen in Table 4.8 the combination of character and word language model achieves the best results for all the task considered. These improvements can be based, in part, on the fact that a character LM can generate sequence of letters that are not real words in the language. The use of a word LM here can prevent this from happening since these incorrect sequences will be given a low probability.

### 4.4.3 Using n-best lists in very constrained scenarios

So far, we have focused on generating text with standard input interfaces (i.e., basically typing on a keyboard). Word or character based approaches have been devised and discussed. However, in some situations, no real typing can be performed and, therefore, different alternatives have to be explored in order to build a suitable interface to introduce text. In general, human evaluation is crucial in this kind of tasks

**Table 4.9:** Probability of occurrence of letters in English. From: Fletcher Pratt, Secret and Urgent: *The Story of Codes and Ciphers Blue Ribbon Books*, 1939, p. 252.

| | | | | | |
|---|---|---|---|---|---|
| A | 0.08151 | J | 0.00132 | S | 0.06101 |
| B | 0.0144 | K | 0.0042 | T | 0.10468 |
| C | 0.02758 | L | 0.03389 | U | 0.02459 |
| D | 0.03788 | M | 0.02536 | V | 0.00919 |
| E | 0.13105 | N | 0.07098 | W | 0.01539 |
| F | 0.02924 | O | 0.07995 | X | 0.00166 |
| G | 0.01994 | P | 0.01982 | Y | 0.01982 |
| H | 0.05259 | Q | 0.00121 | Z | 0.00077 |
| I | 0.06345 | R | 0.06832 | | |

since we aim dealing with a very special case [31],[19]. Anyway, some experiments can be conducted in "laboratory" conditions.

Let's suppose an extreme situation (but realistic for some disabled people) in which only a binary input mechanism can be used (for instance a left or right head or eye movement). For simplicity, we can consider the input interface consisting of two keys. One that allows the user to move within a list of characters and a another one to confirm a selection. Words can be constructed by presenting the user a list containing the alphabet (and, maybe, some punctuation marks) and he or she can choose the desired character by using these keys. From this, we could compute the expected number of movements needed to type each character. This expected value is given by:

$$E(movements) = \sum_{c} \Pr(c) \cdot N(c) \tag{4.11}$$

where $\Pr(c)$ is the probability of the character c and $N(c)$ is the number of movements that the user has to perform to achieve the character $c$ in the list. In Table 4.9 the probability of the English letters is shown.

If the list is alphabetically sorted, the expected value in Eq. (4.11) is 12.74. Clearly, this can be improved by sorting the list according to the letter probabilities obtaining, this way, an expected value of 8.25 for English texts.

The ITG approach can be easily applied here. Initially, we can rely on the character language models described in section 4.4.1 so that we can use this information to properly sort the list, trying to find the optimal order to minimize the number of movements. To summarize, given a context $\mathbf{p}$, the probability $Pr(c|\mathbf{p})$ will be computed for each symbol (letter) $c$ and the list will be sorted based on this probability. We can now compute the average number of movements expected in three different situations.

- An alphabetically sorted list.

**Table 4.10:** Results based on characters list. The size column refers to the number of different characters in each corpora. The results measure the average of the number of movements needed to type a correct character

| Corpus | Size | Movements/Character | | |
|---|---|---|---|---|
| | | Alphab. order | Freq. order | ITG order |
| EUTRANS | 33 | 16.45 | 6.45 | 1.29 |
| XEROX | 55 | 19.28 | 7.16 | 2.03 |
| SHAKESPEARE | 38 | 17.38 | 6.72 | 3.48 |

- A list sorted based on the individual probabilities of the different characters in the alphabet (similar to what it is shown in Table 4.9).
- A dynamic sorted list using the information from the character language models.

These experiments were performed by generating the test sets for several corpora according to each one of these three modalities. As can be observed in Table 4.10, the use of context information derived from the language models can notably improve the performance which entails a very significant reduction of the effort needed to type text in this special scenario.

## 4.5 Adaptive learning

As we discussed in Chapter 2, one of the advantages of the IPR proposal is that during the normal system operation mode, new and completely reliable training material is being produced, which can be used for improving the system performance.

In addition, this new material may not be seen as simple new data but as data heavily related to the task being currently solved. As an specific ITG example, we could develop a system to be used as an e-mail assistant tool and this system could be trained from e-mail samples from different users. However, we can expect more benefit if the system is mainly based on the current user emails. Under an adaptive learning framework, we have at our disposal all the data that the current user is producing. Why do not take advantage of this?

### 4.5.1 Some strategies for adaptive learning

Within ITG, some alternatives to apply an adaptive learning approach can be proposed. Before describing them, we have to say that we will use, in these experiments, only the SHAKESPEARE and EUROPARL corpora since they constitute real, long texts. The others are mere sequences of isolated sentences and, for that reason, are not adequate to be test under an adaptive learning paradigm.

### Cache models

Cache models [28] are well-known models used in speech recognition. The rationale behind these models is the fact that a word seen in a document is likely to occur again in a near future. From this, the cache model increases the probabilities of the words seen in a recent past.

In the adaptive learning approach considered here, a cache model can be useful since the last $M$ validated words in the current task will be probably chosen by the user later on.

Formally, the cache model can be defined as:

$$P_{cache}(v|\mathbf{w}_{n-M}^{n-1}) = \frac{1}{M}\sum_{m=1}^{M}\delta(v, \mathbf{w}_{n-m}) \tag{4.12}$$

where $\delta(v, \mathbf{w}_{n-m}) = 1$ if $\mathbf{w}_{n-m} = v$ and $\delta(\mathbf{w}_{n-m}, v) = 0$ otherwise. To combine the Cache and the $n$-gram models, we will use linear interpolation as in [49]:

$$P(s|p) = P_{n-gram}(\mathbf{s}|\mathbf{p}) \cdot (1-\alpha) + P_{cache}(\mathbf{s}|\mathbf{p}) \cdot \alpha \tag{4.13}$$

In Table 4.11 the results using different interpolation weights are shown.

**Table 4.11:** KSR Results of applying cache on different corpora. A value of 100 has been used for the $M$ parameter. In the final column the baseline results are presented

| Corpus | Factor | | | | | |
|---|---|---|---|---|---|---|
| | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 | base |
| SHAKESPEARE | 42.9 | **42.8** | 43.1 | 44.2 | 46.7 | 43.6 |
| EUROPARL | **27.9** | **27.9** | 28.0 | 28.6 | 30.1 | 28.0 |

With this simple model, only a slight improvement is achieved.

### Interpolating with a unigram model estimated from the test

The second proposal consists in learning a new model from the test data already generated.. Because of the small amount of test sentences, we can not expect to have enough data to properly estimate a complex model. On account of this, a simple unigram distribution has been chosen to summarize the information coming from the test portion already validated.

To combine the new model with the original $n$-gram model, a linear interpolation is proposed again. Let $P_{training}$ be the model off-line estimated from the original training samples and let $P_{test\_unigram}$ be a unigram model estimated from the test samples predicted so far. The probability $P(s|p)$ is computed as:

$$P(\mathbf{s}|\mathbf{p}) = P_{training}(\mathbf{s}|\mathbf{p}) \cdot (1-\alpha) + P_{test\_unigram}(\mathbf{s}) \cdot \alpha \quad (4.14)$$

The model $P_{test\_unigram}$ is updated when a new sentence in the test set has been completely predicted. In Table 4.12 the results of this adaptation technique are summarized.

**Table 4.12:** KSR results of interpolating with a unigram estimated from the test. In the final column the baseline results are presented

| Corpus | $\alpha$ | | | | | |
|---|---|---|---|---|---|---|
| | 0.9 | 0.7 | 0.5 | 0.3 | 0.1 | base |
| SHAKESPEARE | 45.1 | 43.6 | 42.5 | 42.2 | **42.0** | 43.6 |
| EUROPARL | 32.4 | 29.5 | 28.6 | 28.1 | **27.8** | 28.0 |

In the case of SHAKESPEARE more than one point and a half of improvement has been achieved. For EUROPARL, the results are not really significant.

### Feeding the original models with test data

As a final proposal, we are going to consider one of the most direct solutions when dealing with the problem of adaptive learning. The idea is to completely rely on the original models but including into them some information about the domain being solved. To this end, an incremental version of $n$-gram models can be implemented so that only the counts (sufficient statistic) are actually stored in the model. As soon as new material is validated, these counts are updated and, when a prediction is needed, the corresponding probabilities are then computed.

One of the problems arisen in this scenario is caused by the scarce amount of data coming from the task being solved in comparison to the initial training data. Because of this, the impact of the new material is barely significant. In order to solve this problem, a learning factor parameter can be used. This parameter over-scale the test samples seen so far to increase the final contribution of these samples. In Table 4.13 and, due to the fact that no formal methods can be used to determine the over-scaling factors, several values for this parameter have been considered.

**Table 4.13:** KSR Results retraining the models based on a learning factor parameter . In the final column the baseline results are presented

| Corpus | KSR | | | | | |
|---|---|---|---|---|---|---|
| | lf = 1 | lf = 2 | lf = 5 | lf = 10 | lf = 20 | base |
| SHAKESPEARE | 42.1 | **41.9** | 42.0 | 42.1 | 42.4 | 43.6 |
| EUROPARL | 27.3 | 27.3 | 27.1 | **27.0** | 27.1 | 28.0 |

This approximation achieves a better result for EUROPARL. In the case of SHAKESPEARE, the results are similar to those obtained in section 4.5.1.

As a conclusion, the improvements achieved by these adaptive learning techniques are not as high as expected. Nevertheless, this evaluation method may be somewhat unfair and even misleading. In any case, we think this topic deserves a deeper analysis and discussion. Next, these results will be put in context and we will try, as well, to discover the real potential of adaptive learning within ITG.

### 4.5.2 Deeper study of an adaptive learning scenario

In order for the adaptation paradigm to be actually useful, we can (generally) expect an improvement in the models as the amount of adaptation data grows. We are claiming that we can adapt the models to a specific task and, as a result, the user should perceive an increase in the system performance as he or she moves forward in the task. In our experimental framework, this is somehow equivalent to consider the test data as a set of consecutive blocks of sentences and focusing on comparing the evolution of the model performance (with and without adaptation) as we generate these blocks. In other words, we are trying to study how the adapted models evolve when more and more test data is processed. From this, we could achieve reasonable conclusions about whether the user will perceive the system as being more and more accurate or not.

To draw more general conclusions, a new task will be included in the experiments. As was mentioned before, the XEROX corpus consists on sentences from printer manuals. However the sentences in the test set are not completely related to each other. For that reason, a different XEROX test (called XEROX HUMAN-EVAL) will be adopted. This corpus is interesting because it is actually a whole printer manual (and not a mere collection of sentences) and, therefore, constitutes a real scenario to evaluate the adaptive learning approach. The main features of this corpus are shown in Table 4.17.

In Table 4.15 the KSR numbers for this corpus are reported. From this results, we can notice a much better behavior of the adaptation technique in comparison with the other corpora. This can be explained by the fact that XEROX HUMAN-EVAL, is a more constrained task (although completely realistic).

**Table 4.14:** Features of the XEROX HUMAN-EVAL corpus

| | XEROX HUMAN-EVAL partition |
|---|---|
| Training sentences | 51577 |
| Test sentences | 4380 |
| Train vocabulary | 15231 |
| Test running words | 117456 |
| Test running characters | 641721 |
| Test-set perplexity (3-grams) | 60.8 |

**Table 4.15:** Results on the *Xerox human -eval* corpus. The first row shows the results by computing the KSR using only the off-line trained model. In the second row, the results of adapting this trained model with the test material predicted is reported

| | KSR |
|---|---|
| Trained model | 25.1 |
| Off-line Trained model + adaptation | 17.3 |

Before presenting the results we have to take into account an important detail. The way of splitting the whole tests set is crucial to allow for a fair comparison between the two models (the original and the adapted one). The point here is that a bad block-division can introduce some noise since choosing blocks of different *difficulty* can hide the real evolution of the results. As an example, by considering each block as a single sentence, we could not expect a realistic estimation of the evolution of the adapted models since a high variation in the prediction accuracy can be expected from one sentence to the next one.

To deal with this situation, we propose the following solution. Every possible block partition (with some limitations that will be discussed later) will be considered for each test set. These partitions will be scored with some measure that shows the expected *degree of difficulty* of each block. Finally, the partition that proves to be more uniform according to this metric will be chosen. Perplexity is the usual way to measure the *difficulty* of an NLP task. Therefore, the uniformity of a partition is estimated as the variance of the block perplexities. This way, the partition minimizing this variance will be finally chosen.

In Table 4.16 an empirical study about this variance is shown. It is necessary to remark that all the possible partitions have been considered but constraining the block size to be greater than one hundred sentences. Smaller blocks do not deserve to be even considered.

In Figure 4.4 the evolution of the difference between the KSR achieved with the original trained model and with the adaptation technique proposed in section 4.5.1 is represented. As can be observed, these results clearly show that the system performance is constantly increasing as new test data is processed. From this we could infer, to some extent, that adaptive learning is really useful to improve an ITG performance in the medium and long term.

### Learning from scratch

As a final experiment, it could be really interesting to explore a pure adaptation approach. Starting from zero, we can build the statistical prediction model as the test samples are generated (the procedure is identical to what was explained in section 4.5.1, the only difference is that here we start from an empty model). Notice the interest of this approximation since neither model nor training samples are required to build a functional ITG application. At first, the system will be really useless but,

**Table 4.16:** Variance in perplexity based on the number of blocks for XEROX, SHAKESPEARE, EUROPARL and XEROX HUMAN-EVAL corpora. The number represent the variance of the perplexity in the partition as the set is split in different number of blocks

| # blocks in the set | Corpus | | | |
|---|---|---|---|---|
| | XEROX | SHAKESPEARE | EUROPARL | XEROX-HE |
| 2 | **1.54** | 29.2 | 1.7 | 10.2 |
| 3 | 82.2 | 14.5 | **1.1** | 43.4 |
| 4 | 129.9 | **10.5** | 2.3 | 21.7 |
| 5 | 21.9 | 28.4 | 1.9 | **3.1** |
| 6 | 25.9 | 70.8 | 5.1 | 8.9 |
| 7 | 45.0 | 80.2 | 11.4 | 10.3 |
| 8 | 130.1 | 62.5 | 35.3 | 20.4 |
| 9 | 124.3 | 160.9 | 29.0 | 44.9 |
| 10 | 203.4 | 87.6 | 30.9 | 70.8 |

as the user types more and more text, the accuracy is expected to be significantly improved. In order to check the performance of this technique, the XEROX HUMAN-EVAL task used in section 4.5.2 has been tested with this new methodology. In addition, the well known Spanish classical novel "Don Quijote de La Mancha" has been used as an example of a large test set suitable to be used in this kind of experiment.

**Table 4.17:** Features of the DON QUIJOTE corpus

| | DON QUIJOTE |
|---|---|
| Sentences | 36266 |
| Vocabulary | 29227 |
| Running words | 431098 |
| Running characters | 2163177 |

In the case of XEROX HUMAN-EVAL, as can be seen in Table 4.18, the results are worse than those achieved by using a previously trained model plus adaptive learning but better than those obtained using only the initial trained model. For the DON QUIJOTE novel, we can say that, despite being a really difficult task, the KSR figures are very satisfactory. Finally, in Figure 4.5 the evolution of the KSR for this last corpus shown (the whole test has been split into 10 blocks of 3000 sentences and the KSR of each block represents a point in the graph). It is worth noticing how the models can reach a good performance by only processing a small amount of sentences.

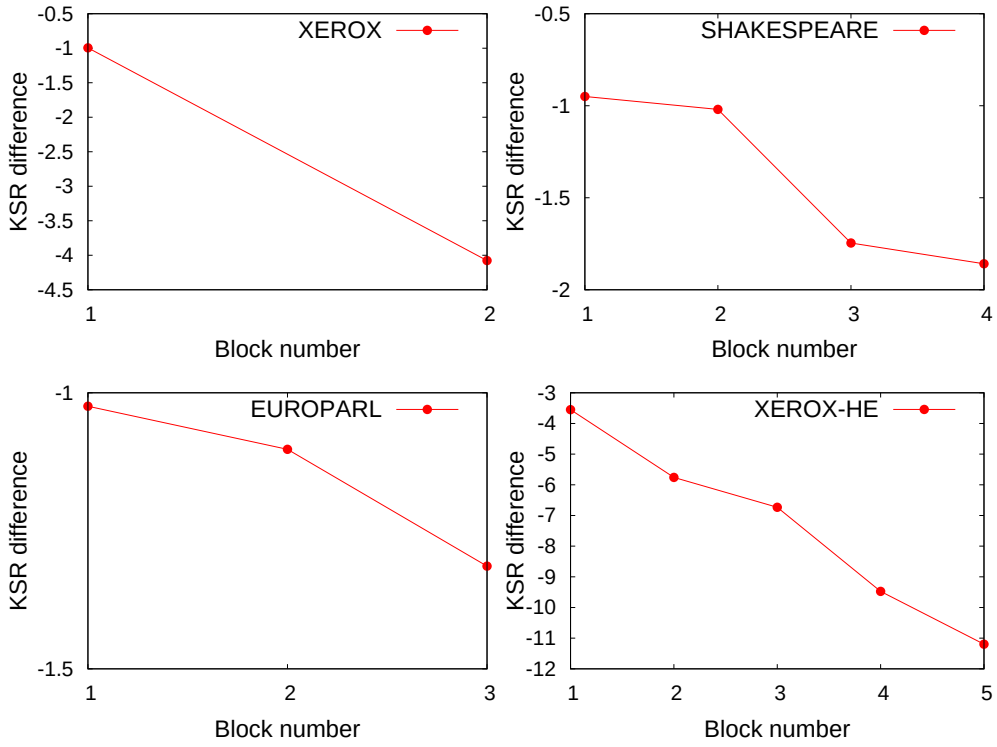



**Figure 4.4:** Evolution of the difference (in KSR) between the adapted model and the original trained model on the four corpora. 2 blocks were considered fox XEROX, 4 blocks for SHAKESPEARE according to Table 4.16, 3 blocks were considered for EUROPARL and 5 blocks for XEROX HUMAN-EVAL according to Table 4.16

**Table 4.18:** Results of a pure adaptive learning approach. The prediction model is constructed *on-the-fly* from the sentences already validated

| | KSR |
|---|---|
| XEROX HUMAN EVAL | 23.8 |
| QUIJOTE | 32.9 |

## 4.6 ITG applications

So far, ITG has been evaluated as a tool to generate documents in natural language. However, this is not the only situation that can be derived from ITG. In this section, a couple of different ITG applications will be described.

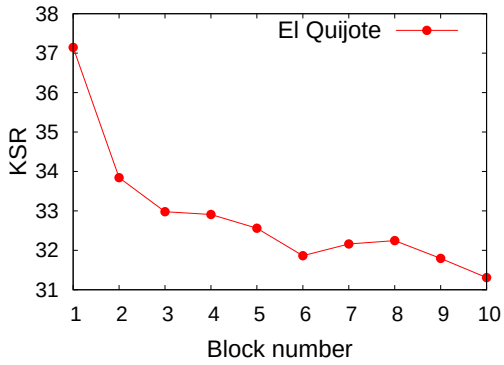



**Figure 4.5:** Evolution of the KSR in the prediction of the novel "Don Quijote de La Mancha" based on the number of test block processed.

### 4.6.1 ITG as a computer-programming assistance tool

Since high-level programming languages were developed, a lot of different tools have been constructed to facilitate the programmer's work. Complex environments for developing software have been devised in the last years and they have become very popular within the software community. Some of these tools present simple ITG features. For instance, automatic completion of types, variables or classes are generally implemented.

The generation of computer programs seems to be a good opportunity to take advantage of the ITG framework. Here, however, automatically generated text portions could be also useful in learning environments, where the user is not really fluent with the language lexicon and syntax. Nevertheless, this claim has to be supported by experiments with real users which is something out of the scope of this work. For that reason, we are going to focus only on the effort reduction in KSR terms that can be achieved. In case that these figures prove a significant prediction accuracy, these human-oriented experiments could be justified.

In this environment, ITG will be assessed by using part of the *Linux* Kernel source code, developed in C language. The Linux kernel consists of different source files grouped into different directories according to the functionality provided. From all these directories, four of them (ARCH, KERNEL, FS and MM) have been chosen so that all their 'C' files will be used as training/test material. In Table 4.19, the main features of the LINUX corpus are presented.

In the experiments, all the source files in the *arch* directory were used as training set and the remaining files were used as test. The KSR results are shown in Table 4.20.

In the first place, we can observe, a worse performance compared to most of the *pure* natural language tasks. This can be somehow surprising since we could expect this task to be easier (notice, however, than no preprocessing techniques were applied here, and the raw files were directly used). On the other hand, $n$-grams could

**Table 4.19:** Features of the LINUX corpus

| | ARCH | KERNEL | FS | MM |
|---|---|---|---|---|
| Number of source files | 316 | 123 | 545 | 55 |
| Running words | 1137470 | 420356 | 2385975 | 239233 |
| Running characters | 6625582 | 2500552 | 14425573 | 1385875 |
| Vocabulary | 43406 | 31403 | 108013 | 17321 |
| Number of lines | 164340 | 61358 | 326582 | 33640 |

**Table 4.20:** Results on the LINUX corpus. The first row shows the results by computing the KSR using only the off-line trained model (trained from ARCH). In the second row, the results of adapting this model with the test material validated so far is reported. Finally, results using the a complete adaptive learning approach as described in the previous sections are also shown

| | FS | KERNEL | MM |
|---|---|---|---|
| Trained model | 56.2 | 53.4 | 53.0 |
| Trained model + adaptation | 40.4 | 41.5 | 43.1 |
| Adaptive learning only | 30.6 | 31.3 | 30.7 |

not be the best models to be considered here. More surprisingly, the application of a full adaptive learning approach achieves the best results, making previously trained models completely useless.

### 4.6.2 ITG and information retrieval

As a final application of ITG, we will consider an interactive information retrieval scenario. In this case, this is not only the application of ITG to a different task, but also a new way of using a text prediction system.

Information Retrieval (IR) systems have gained a lot of importance in daily life. Querying databases is now a usual task and, therefore, appropriate interfaces are crucial to guarantee a real benefit from the huge amount of data currently available. In this sense, interfaces based on natural language have drawn a lot of attention in the last years since they constitute the most natural way for a user to communicate. In the case of accessing to a database this is even more important since the way in which the information is structured can be so complex that other communication alternatives are often useless. Nevertheless, these interfaces are far from being perfect and they present some drawbacks. On the one hand, in spite of allowing the use of natural language, they are usually constrained to a specific vocabulary or grammatical structure. When the input is not constructed following these restrictions (and usually the typical user is not aware of them) the system response is useless and the user normally feels frustrated. On the other hand, typing text can be complicated in some

situations (mobile devices, disabled people, etc.) and this fact can weaken the benefit that a natural language based communication can achieve. This issue is becoming more and more important since nowadays a lot of information is worldwide available and portable devices are becoming the main tool for a significant amount of people.

To overcome these two problems, ITG can be really useful. In the first place, regarding the second drawback, ITG can significantly reduce the effort in terms of interactions (that is key strokes) to generate text in very different tasks. In the second place, concerning the first question, ITG can boost the correct use of an interface based on natural language since ITG can be seen as an interactive user guide to the IR system. This way, ITG can be employed to lead the user to type the input text with the kind of constructions expected by the system and therefore to improve the final results obtained. The point here is that the ITG system could help the user, in the short term, by providing a quick way to obtain an acceptable result for the query currently carried out and, on the other hand, in the long term by showing the kind of queries that the system is more likely to accept. The aim of this section is to roughly explore the possibilities that this proposal could offer on a simple system as well as to discuss an initial experimental framework for this novel application. To this end, a database containing information about train routes will be used in the experiments.

#### Integrating ITG into a natural language based information retrieval system

Different approaches can be taken when including an ITG system into an information retrieval application. Perhaps, the most simple one is based on a completely decoupled architecture where the ITG is used basically as a tool to construct a query hypothesis. In this case, the user interacts with the ITG system as it is shown in Figure 4.2, that is, typing text that is simply auto-completed by the ITG engine. Once the user validates the whole sentence, it is used to query the database. A second alternative arises when considering a loosely-coupled approach. In this case, an ITG system is used as in the previous case but, instead of waiting for the user validation of the whole natural language query, the different text predictions will be used to retrieve information from the database. As a consequence, what the user obtains in each interaction is not merely a text completion but an answer to the query being constructed. Other (and more complex) alternatives can be also explored. For instance, a strongly coupled approach, where the ITG system could be fed with some feedback from the IR process in order to decide which prediction is more convenient.

### 4.6.3 Brief description of the AMSABEL system

In order to assess this initial proposal, a simple information retrieval system called AMSABEL[45] will be employed. AMSABEL is based on the use of statistical machine translation (SMT) techniques to translate from a natural language into a structured query language (SQL). Currently, AMSABEL is able to accept queries both in Spanish and English. The resulting SQL sentences are used to access a railway database where information about train routes is stored (specifically a user can ob-

tain information about departure and arrival cities, starting and ending dates, starting and arrival times, ticket prices, etc.). The translation of the Spanish or English input into SQL is performed by using phrase-based models[27]. These models perform the translation in three steps. Firstly, the input sentence is segmented into phrases (which are sequences of consecutive words). Then, each segment is translated into the corresponding segments in the target language and, finally, the target phrases are properly ordered to achieve the final translation. Formally, in statistical machine translation we are given a source sentence $\mathbf{f}$ , and we try to find the optimal target sentence $\mathbf{e}$ as:

$$\hat{\mathbf{e}} = \underset{\mathbf{e}}{\operatorname{argmax}} \Pr(\mathbf{e}|\mathbf{f}) = \underset{\mathbf{e}}{\operatorname{argmax}} \Pr(\mathbf{f}|\mathbf{e}) \cdot \Pr(\mathbf{e}) \tag{4.15}$$

where $\Pr(\mathbf{e})$ is the language model probability and $\Pr(\mathbf{f}|\mathbf{e})$ is the translation model probability. In the case of phrase models, this probability is expressed as it is stated in Eq. 4.16:

$$\begin{aligned} \Pr(\mathbf{f}|\mathbf{e}) &= \Pr(\bar{\mathbf{f}}_1^I|\bar{\mathbf{e}}_1^I) \\ &\approx \prod_{i=1}^{I} \phi(\bar{\mathbf{f}}_i|\bar{\mathbf{e}}_i) d(\mathbf{a}_i - \mathbf{b}_{i-1}) \end{aligned} \tag{4.16}$$

where $\bar{\mathbf{f}}_i$ is the $i$-th phrase in $\mathbf{f}$, $\bar{\mathbf{e}}_i$ is the $i$-th phrase in $\mathbf{e}$, $\phi(\bar{\mathbf{f}}_i|\bar{\mathbf{e}}_i)$ is the probability of being $\bar{\mathbf{f}}_i$ a translation of $\bar{\mathbf{e}}_i$ and $d(\mathbf{a}_i - \mathbf{b}_{i-1})$ is the distortion model used for reordering target phrases. The order of a target phrase $\bar{\mathbf{f}}_i$ depends on a probability distribution based on its start position ($\mathbf{a}_i$) and the end position of $\bar{\mathbf{f}}_{i-1}$ ($\mathbf{b}_{i-1}$).

Phrase models are obtained from word to word alignments ([5]). The search for the most likely translation is performed by using the *Moses* beam-search decoder [26].

A set of semi-automatically generated input queries was in the experiments (752 English and 748 Spanish sentences were employed) along with the information that the user expects to obtain for each query (see Table 4.21). We think that the use of this synthetic corpus can be justified in this first evaluation stage, since the produced sentences can be, in our opinion, considered as plausible queries for a user interacting with a natural language based IR system. In Figure 4.6 some examples of these test queries are shown.

The evaluation procedure was aimed at directly testing the system usefulness from a potential user point of view. Thus, the results of the input natural language queries were classified based on the outcome obtained (the point here is to measure the system accuracy according to the correctness of the information retrieved). The following query categories were established:

**Q1** *Exact information*: The system provides the user with the exact information required.

**Q2** *More fields*: The system returns the information required but more fields are also provided.

*please , I would like to know which destinations are there from Guadalajara.*
*are the classes of train 1047 ?*
*what times can you go from Toledo to Alicante the 2011-06-03 ?*
*which days of the week can you go to Guadalajara from Ciudad Real ?*
*from which cities can you go to Alicante ?*
*I want to know which destinations there are from Valencia.*

**Figure 4.6:** Examples of semi-automatically generated test sentences

**Q3** *More rows*: The system provides the information required but more rows from the database tables are also shown.

**Q4** *Incorrect*: The query does not include the expected information.

Categories 1 and 2 are completely useful they the user obtain the information that he or she requested (adding in the case of Q2 type more fields). Q3 type can be also considered useful since the information requested is provided although some useless additional information is also included in the result (causing, maybe, some annoyance). In the case of the Spanish test corpus, about 81% of system results were classified as type Q1, less than 0.3% as type 2, about 1.3% as type 3 and, finally, about 17% as type 4 (that is, quite useless).

### 4.6.4 ITG experiments

Once the IR system has been described, a framework to study the possible usefulness of the ITG proposal will be discussed.

Initially, the KSR of a pure ITG approach on the AMSABEL query text corpus was computed and it is shown in the first row of Table 4.22. The idea is to measure the effort required to generate the exact text queries (without actually accesing the database) in the test set. This can be seen, to some extent, as the effort of using a completely decoupled approach in which the system waits for a completely validated text construction and, based on this, a final single access to the database is performed for each query.

However, ITG can be incorporated into an IR system in very different ways (see section 4.6.2). In this work, we will focus on a loosely-coupled architecture. The idea is that, each time the ITG system makes a text prediction, this prediction will become into a query to the database. Once the query has been performed the user will directly validate the database response in the case it is correct or, otherwise, the part of the text prediction that he or she considers error-free (as it is shown in Figure 4.7). Then, a simple correction will be made on this text prediction and the process will be repeated until the user considers the information retrieved as satisfactory. In the

experiments performed here, the user was simulated by the test sets shown in Table 4.21. The effort will be measured in KSR terms.

**Table 4.21:** Test sets used in ITG for information retrieval

| | Spanish | English |
|---|---|---|
| Number of sentences | 748 | 752 |
| Running words | 9413 | 9763 |
| Vocabulary size | 447 | 484 |
| Perplexity (3-grams) | 5.2 | 4.9 |

In the first row of Table 4.22 the results of using ITG on the whole AMSABEL system are shown. These experiments are aimed at measuring the KSR required to obtain one of the three useful query types described in section 4.6.3. This way, the KSR needed to obtain all the queries classified as type Q1 was computed and the same procedure was used for Q2 and Q3 queries (Q4 queries were not considered since they are useless). As can be observed, the Q1 figures show that it is not necessary to generate the exact query to get a perfect result and that some effort can be saved with respect to replicate the input sentence with ITG (as numbers in Table 4.22 shown) and, what is more important, with a small effort in terms of key strokes, useful Q3 queries can be achieved.

**Table 4.22:** ITG and information retrieval results on the AMSABEL corpus

| | Spanish | | English | |
|---|---|---|---|---|
| Query type | KSR | improvement | KSR | improvement |
| Pure ITG | 15.6 | – | 14.7 | – |
| Perfect queries (Q1) | 14.2 | 16 % | 13.0 | 11% |
| More columns (Q2) | 14.2 | 16 % | 12.9 | 12% |
| More rows (Q3) | 10.7 | 36 % | 9.5 | 35% |

## 4.7 Summary of contributions

In this chapter, a new application of the IPR paradigm have been explored. Here, only the user feedback can be used since there is no input pattern to decoded. Different strategies and models have been proposed and evaluated along with several interaction modalities.

A thorough comparison between the traditional maximization of the posterior probability and a new optimal search strategy has been performed. In addition, we have carried out an analysis of the classification errors produced by both modalities in order to show that the interactive scenario can be very different from the general case.

**Iteration 1**

| | |
|---|---|
| *Prediction:* | which days can you go to Madrid from Toledo |
| *SQL:* | *SELECT DISTINCT Dias FROM Tren JOIN Viaje ON Viaje.Tren=Tren.Id_tren WHERE Destino='madrid' AND Origen='toledo'* |
| *Data:* | Train 1012 → Monday, Tuesday, Friday |
| *Prefix:* | **which** |
| *Amendment:* | **which** *c* |

**Iteration 2**

| | |
|---|---|
| *Prediction:* | **which c**lasses can you buy for train number 1040 ? |
| *SQL:* | *SELECT DISTINCT Clase FROM Tren JOIN Billetes ON Billetes.Tren = Tren.Id_tren WHERE N_tren = '1040'* |
| *Data:* | Train 1040 → Tourist, First class |
| *Prefix:* | **which classes can you buy for train number 10** |
| *Amendment:* | **which classes can you buy for train number 10***2* |

**Iteration 3**

| | |
|---|---|
| *Prediction:* | **which classes can you buy for train number 102**8 |
| *SQL:* | *SELECT DISTINCT Clase FROM Tren JOIN Billetes ON Billetes.Tren = Tren.Id_tren WHERE N_tren = '1028'* |
| *Data:* | Train 1028 → Tourist |
| *Prefix:* | **which classes can you buy for train number 102** |
| *Amendment:* | **which classes can you buy for train number 102***9* |

**Iteration 4**

| | |
|---|---|
| *Prediction:* | **which classes can you buy for train number 1029 ?** |
| *SQL:* | *SELECT DISTINCT Clase FROM Tren JOIN Billetes ON Billetes.Tren = Tren.Id_tren WHERE N_tren = '1029'* |
| *Data:* | Train 1029 → Business, First class |

$$KSR = \frac{3}{49} = 0.061 \rightarrow 6.1\%$$

**Figure 4.7:** Example of information retrieval using ITG. In this example, the user tries to obtain information about the different ticket classes for train number 1029. Each time the user makes a key stroke, the system provides a text completion (*Prediction*) which is translated into *SQL* and the corresponding information retrieved is shown (*Data*). In the first three iterations, the information retrieved is not correct and, therefore, the user interacts with the ITG engine correcting the text prediction mistakes. Finally, at iteration 4, the information provided is fully correct and the process ends successfully.

Some adaptive learning techniques have been also proposed. A detailed study of the behavior of adaptive learning within ITG has been also conducted, clearly

supporting the application of these techniques in this specific problem.

Finally, two applications of ITG different from generating pure text documents have been introduced. The use of ITG as a tool to improve the natural language based information retrieval seems to be very promising and, to the best of our knowledge, a completely new approach to this problem.

CHAPTER 5

# MULTI-MODAL INTERACTIVE PATTERN RECOGNITION. A SPEECH INTERFACE FOR COMPUTER ASSISTED TRANSLATION

## 5.1 Multi-modality in IPR systems

As was introduced in chapter 2, the nature of the IPR systems promotes the development of multi-modality as an integral part of the system. This is mainly due to the fact that IPR systems have to deal with two different kind of inputs. On the one hand, we have the input patterns to be recognized and, on the other, the feedback coming from the user. We can cite the example of CAST, where the input pattern is a sequence of spoken words (audio) and the user feedback corresponds to a series of keystrokes and mouse actions. According to this description, multi-modality seems to be only accidental but a multi-modal scenario can be really useful and practical within an IPR system.

The difference in nature between the input pattern and the user feedback mentioned before is only one of the features that boost multi-modality and, indeed, is not the most important one. Actually, there is a crucial aspect in IPR is that the input pattern and the user feedback are somehow related. As we will argue later on we can, to some extent, anticipate the content of a specific user action and, thus, improve the accuracy of the IPR feedback channel with respect to a completely decoupled interface, where the feedback decoder is implemented just using of the self-components.

**A speech and text interface for a Computer Assisted Translation System**

Within the IPR paradigm, one of the most explored tasks is Computer Assisted Translation (CAT). Translation is nowadays an important activity in many official institutions (EU parliament, the Canadian Parliament, UN sessions, Catalan and Basque parliaments in Spain, etc.) and private companies (user's manuals, newspapers, etc.). The idea behind those CAT systems is to use a Machine Translation (MT) system to produce portions of the target sentence that can be accepted or amended by a human translator. These correct portions are then used by the MT system to produce further, hopefully improved suggestions.

In the interactive systems described in this work, the user usually relies on traditional input methods (keyboard, mouse, etc.) to send the feedback to the system. We present in this chapter an alternative to this idea within the CAT framework. In this proposal, the human translator determines acceptable prefixes of the suggestions made by the system by reading (with possible modifications) parts of these suggestions. With respect to using a general purpose ASR, the IPR framework allows for a much lowwer freedom degree. As we will see, the corresponding lower perplexity would allow for sufficiently high recognition accuracy. Moreover, as this is fully integrated within the CAT paradigm, the user can make use of the conventional means (keyboard and/or mouse) to guarantee that the produced text exhibits an adequate level of quality. Preliminary empirical results, presented in this chapter, support the potential usefulness of using speech within the CAT paradigm.

A significant part of the work presented in this chapter can be found in [51], where the speech interface for CAT is discussed within the context of Speech-to-Speech machine translation.

## 5.2 Introduction to machine translation

The statistical framework for MT can be stated as follows: Given a sentence $\mathbf{f}$ from a source language, search for a sentence from a target language $\hat{\mathbf{e}}$ for which the posterior probability is maximum, that is:

$$\hat{\mathbf{e}} = \underset{\mathbf{e}}{\operatorname{argmax}} \Pr(\mathbf{e} \mid \mathbf{f}) \ . \tag{5.1}$$

As in the general Pattern Recognition case, it is commonly accepted that a convenient way to deal with Eq. (5.1) is to transform it by using the Bayes' theorem:

$$\hat{\mathbf{e}} = \underset{\mathbf{e}}{\operatorname{argmax}} \Pr(\mathbf{e}) \cdot \Pr(\mathbf{f} \mid \mathbf{e}) \ . \tag{5.2}$$

important role in T2TT. On the one hand, $\Pr(\mathbf{e})$ is modeled by a *language model* which gives high probability to well formed target sentences. Smoothed $n$-grams [9] are often used for these language models. On the other hand, models for $\Pr(\mathbf{f} \mid \mathbf{e})$ should give high probability for those sentences from the source language which are good translations for a given target sentence. These models generally consist

of *stochastic dictionaries*, along with adequate models to account for *word alignments* [5, 6, 34] (the concept of word alignment will be described later). An alternative to Eq. (5.2) is to transform Eq. (5.1) as:

$$\hat{\mathbf{e}} = \underset{\mathbf{e}}{\operatorname{argmax}} \Pr(\mathbf{f}, \mathbf{e}) \ . \tag{5.3}$$

In this case, the joint probability distribution can be adequately modeled by means of *stochastic finite-state transducers (SFST)* [7, 8] among other possible models.

## 5.3 Computer-Assisted Translation

As in CAST we can take advantage of the IPR framework described in Chapter 1 to increase the productivity of the whole translation process (MT plus human work) by incorporating the human correction activities within the translation process itself [29]. The idea is to use a MT system to produce portions of the target sentence that can be accepted or amended by a human translator and these correct portions are then used by MT system as additional information to achieve further, hopefully improved suggestions. This approach is followed in the CAT systems presented in [12, 10, 14, 13]. Formally, we have the source sentence $\mathbf{f}$ and a prefix of the translation $\mathbf{e}_p$ and we have to find an optimal suffix according to Eq (5.4)

$$\hat{\mathbf{e}}_s = \underset{\mathbf{e}_s}{\operatorname{argmax}} \Pr(\mathbf{e}_s \mid \mathbf{f}, \mathbf{e}_p) \ . \tag{5.4}$$

Alternatively, we can rely on the joint probability of the source and target sentence which leads to Eq (5.5):

$$\hat{\mathbf{e}}_s = \underset{\mathbf{e}_s}{\operatorname{argmax}} \Pr(\mathbf{e}_p, \mathbf{e}_s, \mathbf{f}) \ . \tag{5.5}$$

### 5.3.1 Speech Recognition for Computer-Assisted Translation

As was commented in section 5.1, IPR systems provide an adequate scenario to develop multi-modality. In the case of a CAT system, the translation productivity could be further increased if speech is used as part of a multi-modal interface. There are several ways to use target-language speech recognition in a CAT system. From free dictation decoding to a very constrained speech recognition.

#### Free target language dictation

As it was already explored in [17, 3, 4] and more recently in [22, 23, 44], the idea is that a human translator *dictates* the translation of a given source text. Consequently, we have a speech decoding process for the *target language* in which knowledge about the source text can be used to attempt reducing recognition errors. Formally, if $\mathbf{f}$ is the given source text and $\mathbf{x}$ is the acoustic sequence corresponding to a target-language utterance produced by a human translator, the search problem becomes:

$$\hat{\mathbf{e}} = \underset{\mathbf{e}}{\operatorname{argmax}} \Pr(\mathbf{e} \mid \mathbf{f}, \mathbf{x}) \;, \tag{5.6}$$

and, assuming that $\Pr(\mathbf{x} \mid \mathbf{f}, \mathbf{e})$ does not depend on $\mathbf{f}$,

$$\hat{\mathbf{e}} = \underset{\mathbf{e}}{\operatorname{argmax}} \Pr(\mathbf{e} \mid \mathbf{f}) \cdot \Pr(\mathbf{x} \mid \mathbf{e}) \tag{5.7}$$

$$= \underset{\mathbf{t}}{\operatorname{argmax}} \Pr(\mathbf{f} \mid \mathbf{e}) \cdot \Pr(\mathbf{e}) \cdot \Pr(\mathbf{x} \mid \mathbf{e}) \;. \tag{5.8}$$

In both equations, $\Pr(\mathbf{x} \mid \mathbf{e})$ can be approached by means of *(target) acoustic models* such as HMMs as in conventional ASR. In addition, to make use of the first equation, $\Pr(\mathbf{e} \mid \mathbf{f})$ can be implemented as a *special (target) language model* that takes into account restrictions derived from the fact that $\mathbf{e}$ has to be a translation of $\mathbf{f}$. The second equation, on the other hand, requires a *translation model* to approach $\Pr(\mathbf{f} \mid \mathbf{e})$ and a *conventional target language model* to approach $\Pr(\mathbf{e})$.

While this is certainly an interesting framework, perfect recognition does not seem possible, even using the information from the source-language sentences (as it is corroborated in the results presented in [17, 3, 4, 22, 23, 44] and our results of section 5.4.6 suggest). Therefore a significant human error-correcting effort would still be required. Moreover, human translators can also make errors themselves, specially when the translation is dictated rather than typed.

### 5.3.2 Speech decoding framework in the CAT paradigm

As an alternative to pure dictated translation we propose a framework that, on the one hand fits well within the CAT paradigm (human correction activities embedded into the process) and, on the other, allows for higher speech recognition accuracy by using lower-perplexity models in the speech recognition process.

Under the CAT framework, a user-validated prefix and a system hypothesis to complete this prefix are available at each stage of the human-system interaction process. When a certain level of accuracy is achieved, this hypothesis is often acceptable or close to it. Therefore, rather than allowing quasi-free dictation (only constrained by the source-text), it is expected that the user dictates text which is an adequate continuation of the preceding prefix and is restricted to be (very) close to the given completion hypothesis. Of course, by sticking to the CAT paradigm, system hypotheses should always be human-amendable. An important difference with respect to the text-only version of CAT discussed in section 5.3, is that now the human can both type and/or speak. The key point is that speech should be encouraged only if low-perplexity recognition is possible, while typing should be preferred when the results of speech recognition are expected to be poor.

As in section 5.3, let $\mathbf{f}$ be the source text and $\mathbf{e}_p$ a validated prefix of the target sentence. The user is then allowed to utter some words ($\mathbf{x}$) generally related to the suffix suggested by the system in the previous iteration, aimed at accepting or correcting parts of this suffix and/or adding some text. Moreover, the user may type some keystrokes ($\mathbf{k}$) in order to correct (other) parts of this suffix and/or to add more text.

| | | |
|---|---|---|
| **ITER-0** | ($\mathbf{e}_p$) | ( ) |
| **ITER-1** | ($\hat{\mathbf{e}}_s$) | (*Haga clic para cerrar el diálogo de impresión*) |
| | (**x**) |  |
| | ($\hat{\mathbf{d}}$) | (**Haga clic a**) |
| | (**k**) | (**en ACEPTAR**) |
| | ($\mathbf{e}_p$) | (Haga clic en ACEPTAR) |
| **ITER-2** | ($\hat{\mathbf{e}}_s$) | (*para cerrar el diálogo de impresión*) |
| | (**x**) |  |
| | ($\hat{\mathbf{d}}$) | (**cerrar el cuadro**) |
| | (**k**) | ( ) |
| | ($\mathbf{e}_p$) | (Haga clic en ACEPTAR para cerrar el cuadro) |
| **FINAL** | ($\hat{\mathbf{e}}_s$) | (*de diálogo de impresión*) |
| | (**k**) | (**#**) |
| | ($\mathbf{e}_p \equiv \mathbf{e}$) | (**Haga clic** en ACEPTAR para **cerrar el cuadro** de diálogo de impresión) |

**Figure 5.1:** Example of typewriter and speech interaction with a CAT system, to translate the English sentence "Click OK to close the print dialog". Each iteration starts with a target language prefix $\mathbf{e}_p$ that has been fixed in the previous iteration. First, the system suggests a suffix $\hat{\mathbf{e}}_s$ and then, the user speaks (**x**) and/or types some key-strokes (**k**), possibly aimed to amend $\hat{\mathbf{e}}_s$ (and maybe $\hat{\mathbf{d}}$). A new prefix, $\mathbf{e}_p$, is built from the previous prefix, along with (parts of) the system suggestion, $\hat{\mathbf{e}}_s$, the decoded speech, $\hat{\mathbf{d}}$, and the typed text in **k**. The process ends when the user types the special keystroke "**#**". System suggestions are printed in cursive, text decoded from user speech in boldface and typed text in boldface typewriter font. In the final translation, **e**, text obtained from speech decoding is marked in boldface, while typed text is underlined.

Using these informations, the system has to suggest a new suffix $\hat{\mathbf{e}}_s$ as a continuation of the previous prefix, the decoded speech and the typed text. That is, the problem would be to find $\hat{\mathbf{e}}_s$ given $\mathbf{f}, \mathbf{e}_p, \mathbf{x}$ and $\mathbf{k}$, considering all possible decodings of **x** (i.e., letting the decoding of **x** be a hidden variable).

According to this very general discussion, it might be assumed that the user can type with independence of the result of the speech decoding process. However, it can be argued that this generality is not realistically useful in practical situations. Instead, it is much more natural that the user waits for a system outcome ($\hat{\mathbf{d}}$) from the spoken utterance, prior to start typing amendments (**k**) to the (remaining part of the previous) system hypothesis. Furthermore, this allows the user to fix possible speech recognition errors in $\hat{\mathbf{d}}$.

In this more realistic and simpler scenario, an alternative problem can be formulated in two steps, as illustrated in Figure 5.1. The first step is to rely on the source text $\mathbf{f}$ and the previous target prefix $\mathbf{e}_p$, in order to search for a target suffix $\hat{\mathbf{e}}_s$:

$$\hat{\mathbf{e}}_s = \operatorname*{argmax}_{\mathbf{e}_s} \Pr(\mathbf{e}_s \mid \mathbf{f}, \mathbf{e}_p) \; . \tag{5.9}$$

This equation exactly corresponds to the CAT scenario discussed in section 5.3 and it can be approached using the same techniques already mentioned in that section (see, e.g., [39, 12]).

Once $\hat{\mathbf{e}}_s$ is available, the user can produce some speech, **x**, and the system has to

decode $\mathbf{x}$ into a target sequence of words, $\hat{\mathbf{d}}$:

$$\hat{\mathbf{d}} = \underset{\mathbf{d}}{\operatorname{argmax}} \Pr(\mathbf{d} \mid \mathbf{f}, \mathbf{e}_p, \hat{\mathbf{e}}_s, \mathbf{x}) \; . \tag{5.10}$$

Finally, the user can enter adequate amendment keystrokes $\mathbf{k}$, if necessary, and produce a new consolidated prefix, $\mathbf{e}_p$, based on the previous $\mathbf{e}_p$, $\hat{\mathbf{d}}$, $\mathbf{k}$ and parts of $\hat{\mathbf{e}}_s$.

We focus now on different manners to approach Eq. (5.10). To start with, we can write:

$$\underset{\mathbf{d}}{\operatorname{argmax}} \Pr(\mathbf{d} \mid \mathbf{f}, \mathbf{e}_p, \hat{\mathbf{e}}_s, \mathbf{x}) = \underset{\mathbf{d}}{\operatorname{argmax}} \Pr(\mathbf{d} \mid \mathbf{f}, \mathbf{e}_p, \hat{\mathbf{e}}_s) \cdot \Pr(\mathbf{x} \mid \mathbf{f}, \mathbf{e}_p, \hat{\mathbf{e}}_s, \mathbf{d}) \; . \tag{5.11}$$

and, by making the reasonable assumption that $\Pr(\mathbf{x} \mid \mathbf{f}, \mathbf{e}_p, \hat{\mathbf{e}}_s, \mathbf{d})$ only depends on $\mathbf{d}$:

$$\hat{\mathbf{d}} = \underset{\mathbf{d}}{\operatorname{argmax}} \Pr(\mathbf{d} \mid \mathbf{f}, \mathbf{e}_p, \hat{\mathbf{e}}_s) \cdot \Pr(\mathbf{x} \mid \mathbf{d}) \; . \tag{5.12}$$

$\Pr(\mathbf{x} \mid \mathbf{d})$ can be modeled by the acoustic models of the words in $\mathbf{d}$ and $\Pr(\mathbf{d} \mid \mathbf{f}, \mathbf{e}_p, \hat{\mathbf{e}}_s)$ can be provided by a target language model constrained by the previous prefix $\mathbf{e}_p$, by the source sentence $\mathbf{f}$ and by the suffix $\hat{\mathbf{e}}_s$ produced at the beginning of the current iteration.

As will be discussed in section 5.4.6, Eq. (5.12) does not lend itself for simple experimentation under laboratory conditions. Therefore, we consider two simplifications that allow for adequate experimentation and are useful in practice too.

First, a *less restricted* scenario arises if only the prefix $\mathbf{e}_p$ is available ($\hat{\mathbf{e}}_s$ is not used). That is, the previous system prediction is ignored and the user is assumed to produce free target speech, only constrained to be a translation of the source text and a continuation of the given prefix:

$$\hat{\mathbf{d}} = \underset{\mathbf{d}}{\operatorname{argmax}} \Pr(\mathbf{d} \mid \mathbf{f}, \mathbf{e}_p) \cdot \Pr(\mathbf{x} \mid \mathbf{d}) \; . \tag{5.13}$$

As compared with Eq. (5.7) of the dictated-translation framework, Eq. (5.13) adds the constraint provided by the target text prefix, $\mathbf{e}_p$, thereby allowing for higher speech decoding accuracy. Along with Eq. (5.13), this alternative can be considered as a combination of the text-only CAT systems of [39] or [12] and the target language dictation systems introduced in [17, 3, 4].

On the other hand, in a *most restricted* scenario for Eq. (5.12), the decoding of $\mathbf{x}$ is constrained to be *exactly* a prefix of the suffix suggested by the system, $\hat{\mathbf{e}}_s$. The idea is that the uttered prefix would help the user determine an accepted part of the system suggestion. In this case, $\Pr(\mathbf{d} \mid \mathbf{f}, \mathbf{e}_p, \hat{\mathbf{e}}_s) = \Pr(\mathbf{d} \mid \hat{\mathbf{e}}_s)$ and Eq. (5.12) can be written as:

$$\hat{\mathbf{d}} = \underset{\mathbf{d}}{\operatorname{argmax}} \Pr(\mathbf{d} \mid \hat{\mathbf{e}}_s) \cdot \Pr(\mathbf{x} \mid \mathbf{d}) \; . \tag{5.14}$$

As compared with all the previous scenarios involving speech, here $\Pr(\mathbf{d} \mid \hat{\mathbf{e}}_s)$ can be modeled by a very low perplexity language model, which should allow for much higher speech decoding accuracy.

## 5.4 Implementing speech decoding in a CAT system

Preliminary versions of CAT systems using target-language speech have been implemented for the two last scenarios described in section 5.3.1. In addition, two extra *"baseline"*, pure *speech recognition* scenarios have been considered, where the information provided by the source-language text is ignored. The goal of these scenarios is to study how speech decoding performance within CAT can be improved by introducing translation constraints.

### 5.4.1 DEC scenario

The least constrained setting is referred to as DEC. It involves just conventional speech decoding of utterances (**x**) of *fragments* of the target sentence:

$$\hat{\mathbf{d}} = \underset{\mathbf{d}}{\operatorname{argmax}} \Pr(\mathbf{d}) \cdot \Pr(\mathbf{x} \mid \mathbf{d}) \; . \tag{5.15}$$

The language model for $\Pr(\mathbf{d})$ is implemented as a (smoothed) $n$-gram, estimated from the same target sentences used to estimate the translation models for other scenarios. Since the $n$-gram is estimated from complete target sentences, but $\mathbf{x}$ is typically an utterance of a sentence fragment, this language model has to be adapted to properly accept any possible subsequence of words.

DEC is more difficult than the translation dictation setting discussed in section 5.3.1, not only because DEC does not take advantage of the information provided by the source-language text, but also because it deals with the decoding of fragments of the target sentence, rather than whole sentences as in [17, 3, 4].

Regarding the changes needed to implement the DEC scenario, it is necessary to remark that an $n$-gram model is intended to score full sentences in a language. In our case, we have a text prefix and a speech utterance that is a partial continuation to this prefix (that is, the prefix and the speech utterance does not form a whole sentence but a longer prefix of a whole sentence). This fact poses two different problems to be solved. The first one is related to the end-of-sentence probabilities. In a traditional $n$-gram, a special token is used to denote an end-of-sentence event so that the recognizer is typically forced to consider as final hypothesis only those ones containing this special event. From a statistical point of view, this entails that those words placed at the end of the training sentences are more likely to be ending words. This is perfectly reasonable when decoding full sentences but not when the utterances are sentence fragments. An easy and effective solution for this problem can be reached by removing these end-of-sentence events from the training set. As a result, all the words are equally likely to be final words in the hypothesis.

### 5.4.2 DEC-PREF scenario

The second pure speech-decoding scenario is referred to as DEC-PREF. Now the available prefix $\mathbf{e}_p$ is introduced as an additional constraint; but, again, no information

about the source text is used. In this case,

$$\hat{\mathbf{d}} = \underset{\mathbf{d}}{\operatorname{argmax}} \Pr(\mathbf{d} \mid \mathbf{e}_p) \cdot \Pr(\mathbf{x} \mid \mathbf{d}) \; . \tag{5.16}$$

To implement DEC-PREF, the same approach described above for DEC have to be followed in order to properly deal with the end-of-sentence problem. On the other hand, we have to address the adaptation of the language model to take advantage of the text prefix. This problem is was already addressed in CAST (Chapter 3). As a quick reminder, we have to decode a partial sentence that is a continuation of this prefix and, therefore, the search should start as if the language model were just observed the prefix.

### 5.4.3 CAT-PREF scenario

The least constrained CAT scenario is called CAT-PREF. It corresponds to a realization of Eq. (5.13), where the source sentence, $\mathbf{f}$, the previous prefix, $\mathbf{e}_p$, and a human-translator utterance, $\mathbf{x}$, are available. The goal of the CAT system is to decode $\mathbf{x}$ into an optimal $\hat{\mathbf{d}}$ and to produce a suggested suffix $\hat{\mathbf{e}}_s$ as a continuation of this decoding.

The combination of CAT-PREF and the CAT-SEL scenario described in section 5.4.4 would allow for pure-speech interactions within a CAT system. CAT-SEL, as will be described, can be successfully solved by using a special and very constrained language model.

On the contrary, CAT-PREF is, maybe, the most interesting scenario, since the baseline results (DEC and DEC-PREF) leave sufficient margin to improve the speech recognition accuracy. Moreover, including the information coming from the source sentence into the speech recognition process constitutes a really challenging scientific problem.

We can see Eq. (5.13) as a ASR problem, where we have an acoustic model, $\Pr(\mathbf{x} \mid \mathbf{d})$, and a language model, $\Pr(\mathbf{d} \mid \mathbf{f}, \mathbf{e}_p)$. The only difference is that this language model is conditioned to the prefix $\mathbf{e}_p$ and to the source sentence $\mathbf{f}$. We have already described how to cope with the prefix. Regarding the source sentence we can adapt the model probabilities so that the weight of each $n$-gram, $P(\mathbf{e}_i \mid \mathbf{e}_{i-n+1}^{i-1})$ is multiplied by the largest probability of translating $\mathbf{e}_n$ from any source word in $\mathbf{f}$; i.e., $\max_{1 \leq j \leq |\mathbf{f}_j|} P(\mathbf{f}_j \mid \mathbf{e}_i)$, where $|\mathbf{f}|$ is the number of words in $\mathbf{f}$. This can be seen as a simple interpretation of the inverted alignments model described in [34]. The lexical translation probabilities $P(\mathbf{f}_j \mid \mathbf{e}_n)$ are obtained from a stochastic dictionary estimated using a parallel corpus and the GIZA++ toolkit [38].

Since we are dealing with a smoothed $n$-gram, in order for the resulting re-scored model to be a real probability distribution we could, initially, act only on the unigram party to posteriorly normalize the results. Although better alternatives are described below, we will include this scenario (called UCAT-PREF) in the results for informative purposes.

As a second alternative, all the $n$-grams in the model can be properly considered by paying careful attention to the underlying smoothing in the $n$-gram model. This

smoothing entails discounting some probability mass from each $n$ order distribution to be transfered to lower order distributions. In order to preserve the discount coefficients computed during the $n$-gram training process we will carry out the following normalization method. Let $h_1 \dots h_{n-1}$ be the history for the $n$-grams to be currently rescored. We can then compute the cumulative probability for these $n$-grams as $acum(h_1 \dots h_{n-1}) = \sum_h P(h|h_1 \dots h_{n-1})$ and use this as a normalization factor after re-scoring with a statistical dictionary. As a result, the only effect of this process is to differently distribute this mass according to a statistical dictionary. An example is depicted in Figure 5.2

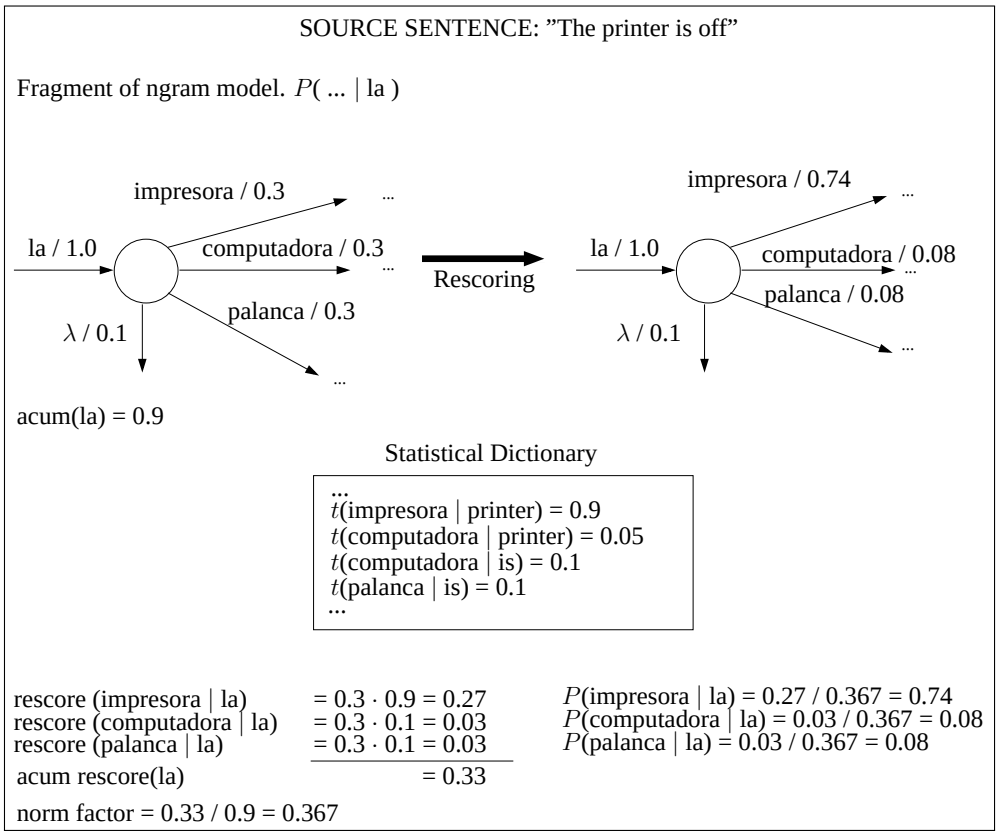



**Figure 5.2:** Example of $n$-gram rescoring. A fragment of a smoothed bigram model is represented as a finite state grammar ($\lambda$ denotes the mass probability discounted from this bigram). Initially, the cumulative probability for all the events is computed (0.9). Then each bigram probability is recomputed according to the statistical dictionary and the source sentence (the maximum probability for each pair source/target word is used). Finally, the new probability distribution is obtained by normalizing on the previous and new cumulative probabilities.

A second alternative arises by rewriting Eq. (5.13) as it is shown in Eq. (5.17).

$$\begin{aligned}\hat{\mathbf{d}} &= \operatorname*{argmax}_{\mathbf{d}} \Pr(\mathbf{d} \mid \mathbf{f}, \mathbf{e}_p) \cdot \Pr(\mathbf{x} \mid \mathbf{d}) \\ &= \operatorname*{argmax}_{\mathbf{d}} \Pr(\mathbf{f} \mid \mathbf{e}_p, \mathbf{d}) \cdot \Pr(\mathbf{d} \,|\mathbf{e}_p) \cdot \Pr(\mathbf{x} \mid \mathbf{d}) \,. \end{aligned} \tag{5.17}$$

As a result we have, on the one hand the previously described acoustic model $\Pr(\mathbf{x} \mid \mathbf{d})$, a prefix-conditioned language model $\Pr(\mathbf{d} \mid \mathbf{e}_p)$ and a whole translation model $Pr(\mathbf{f} \mid \mathbf{e}_p)$.

Directly approaching the maximization in Eq. (5.17) using the usual dynamic programming techniques seems to be not feasible. However, word graphs or $n$-best list from speech decoding allows for a straightforward implementation of Eq. (5.17). A statistical translation model can be here applied to score a set of ASR hypotheses This approach to CAT-SEL scenario is called IBMCAT-PREF.

Five different models (called IBM 1,2,3,4 and 5 models) have been devised so far and they are based on the concept of word alignment, which represents a mapping between the words in the source sentence and their corresponding translations in the target sentence. Models IBM 3, IBM 4 and IBM 5 are considerably more complex than 1 and 2 and, therefore, finding an optimal alignment becomes an NP-complete problem [24] where only approximate solutions can be followed. We we are going to rely, in this case, on the IBM model 1, which computes the probability of having the source sentence $\mathbf{f}$, given the target $\mathbf{e}$ sentence and a word-to-word alignment $\mathbf{a}$ as it is defined in Eq (5.18):

$$p(\mathbf{f}|\mathbf{a}, \mathbf{e}) = \frac{\epsilon}{(|\mathbf{e}|+1)^{|\mathbf{f}|}} \prod_{j=1}^{|\mathbf{f}|} t(\mathbf{f}_j|\mathbf{e}_{\mathbf{a}_j}) \tag{5.18}$$

where $t$ denotes a statistical dictionary and $\mathbf{a}$ is a vector of size $|\mathbf{f}|$ representing the word alignments ($e_{a_j}$ represents the word in the target sentence that is aligned with the $j$-th word in the source sentence). Finally, $\epsilon$ represents the probability $Pr(|\mathbf{f}| \mid \mathbf{e})$ (that is, a uniform distribution for the source length).

Usually, from all the possible alignments , only the most likely one (called *Viterbi* alignment) is considered and, hence, $Pr(\mathbf{f}|\mathbf{e})$ is approximated by $Pr(\mathbf{f}|\hat{\mathbf{a}}, \mathbf{e})$. The *Viterbi* alignment can be efficiently computed in Model 1 by aligning each word in the source sentence to the word in the target sentence that maximizes the score for the partial alignment constructed so far (IBM models constrained the alignments so that each word in the source sentence can be aligned as most to one word in the target sentence). In Figure 5.3 an example of the computation of this optimal alignment is shown.

IBM models, however, are trained to cope with full sentences and, in this scenario, we have a full source sentence and a fragment of a target sentence. As a result, the model can produce some *artificial* alignments in order to deal with this unexpected situation an this could result in the achievement of unreliable scores. Different possibilities arise to address this problem. Initially, we could take advantage of the

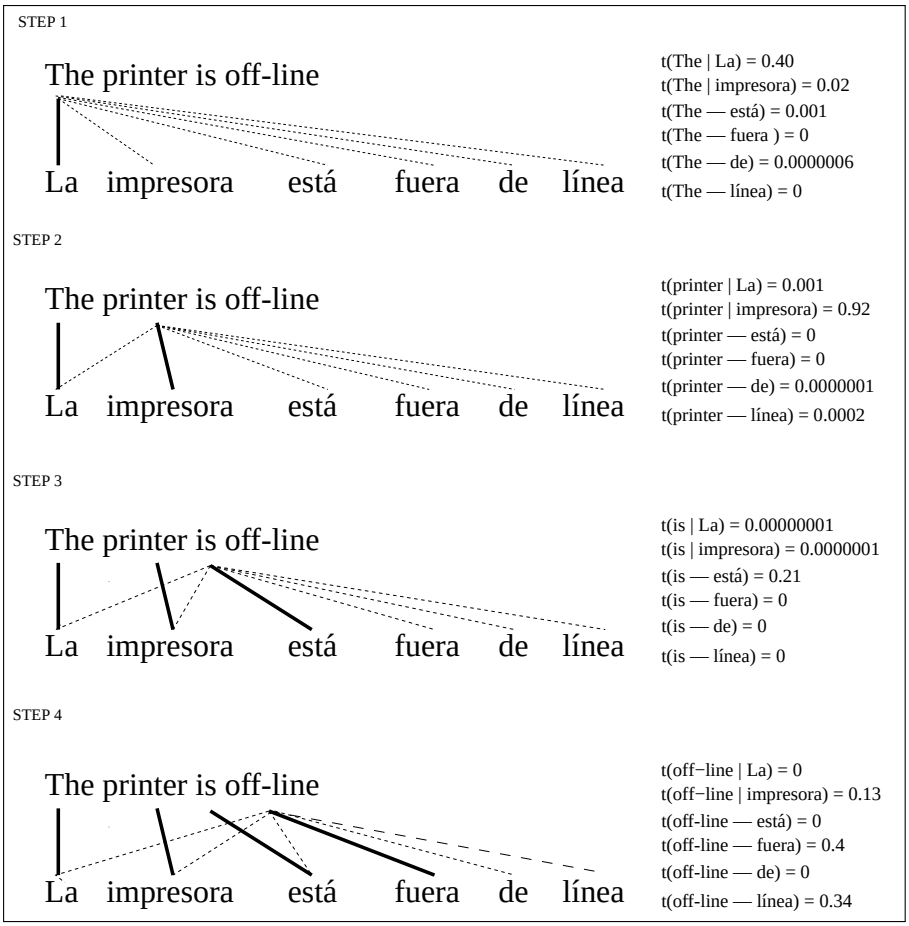



**Figure 5.3:** Example of IBM model 1 alignment computation. Each word in the source sentence (*The printer is off-line*) is aligned to the optimal word in the target sentence (*La impresora está fuera de línea*). The boldfaced lines represent the optimal alignment for the current source word chosen at each step whereas the dotted lines represent the rest of alignments considered and finally discarded.

CAT scenario in which the system is embedded so that the CAT engine could provide the best possible completion for the each utterance hypothesis. However this can be clearly prohibitive, since the number of the hypotheses to be completed will be, in general, too high.

On the other hand, as was mentioned before, the IBM model alignments are constrained so that a word in the source sentence can be aligned to as most to one word in the target sentence. When having a full source sentence and a target fragment, the model can try to align every word in the source sentence to a word in the fragment (even when some source words may not be related to any fragment word). We can see that, here, this alignment structure is working against us. Indeed, the kind of

constraints that could help us to overcome this problem are provided by the model trained in the opposite direction, that is, $P(\mathbf{e}_t, \mathbf{d} \mid \mathbf{f})$. From this model point of view, each word in the target fragment would be aligned to the best word in the source sentence and, consequently, those source words that are not real translations of the words in the fragment (that is, words not translated yet) can be discarded. In Figure 5.4, an example of this is shown.

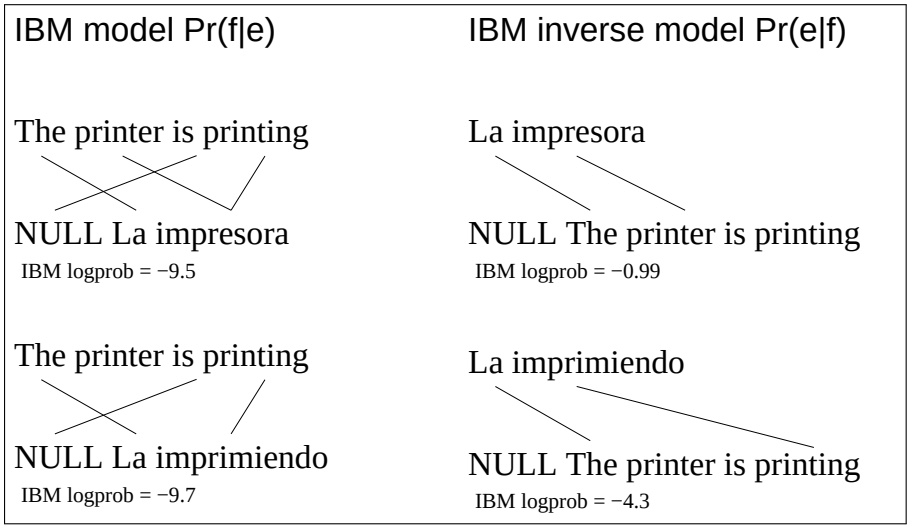



**Figure 5.4:** Example of forced alignment solved by using the inverse model. The real utterance was "La impresora". However, the model $P(\mathbf{f}|\mathbf{e})$ forces an alignment between the words "impresora" and "printing". As a consequence a wrong translation "La imprimiendo" is scored with higher probability than the correct translation fragment. The use of the $P(\mathbf{e}|\mathbf{f})$ model can contribute to solve this problem.

We can benefit from the fact that state-of-the-art statistical machine translation relies on log-linear models to approach the translation probability. The main motivation is to properly combine different information sources (models), as it is shown in Eq (5.19):

$$P(\mathbf{f} \mid \mathbf{e}_t, \mathbf{d}) \propto exp\left[\sum_i \lambda_i f_i(\mathbf{f}, \mathbf{e}_t, \mathbf{d})\right] \tag{5.19}$$

where $\mathbf{f}_i$ are the functions to be combined in the model and $\lambda_i$ are the weights. In our case, can use two different functions, the original $P(\mathbf{f} \mid \mathbf{e}_t, \mathbf{d})$ used in Eq.(5.17) and the $P(\mathbf{e}_t, \mathbf{d} \mid \mathbf{f})$ aimed at improving the alignment quality. The $\lambda_i$ weights will be optimized according to a minimum error rate training criterion on a development set. This approach to CAT-PREF scenario is called IBMLLCAT-PREF.

### 5.4.4 CAT-SEL scenario

Finally, the most constrained scenario, called CAT-SEL, corresponds to a realization of Eq. (5.14). It is similar to CAT-PREF but here the human translator can only utter exact prefixes of the suggestion made by the CAT system ($\hat{\mathbf{w}}_s$). These utterances are aimed at *selecting* acceptable prefixes of the system suggestions (hence the name, CAT-SEL). The possible amendments of the remaining parts of the suggestions can only be made by typing (or by applying CAT-PREF). In practice, Eq. (5.14) can be implemented as a search for $\hat{\mathbf{d}}$ in the (small) set of possible prefixes of the target suffix $\hat{\mathbf{e}}_s$; that is, $\Pr(\mathbf{d} \mid \hat{\mathbf{e}}_s)$ is estimated by a special finite-state language model in which only those $\mathbf{d}$ that are prefixes of $\hat{\mathbf{e}}_s$ have non-null probability. The acoustic models for estimating $\Pr(\mathbf{x} \mid \mathbf{d})$ in Eq. (5.14) are the same HMMs as in all the previous cases.

The computation of $\hat{\mathbf{e}}_s$ in Eq. (5.9) for CAT-PREF and CAT-SEL is the conventional optimization of text-only CAT, which can be solved by Eq. (5.4) or Eq. (5.5) (see [39, 12]). In CAT-PREF, the prefix $\mathbf{e}_p$ is the concatenation of the old $\mathbf{e}_p$ and $\hat{\mathbf{d}}$ (and $k$ if the user typed some text). In CAT-SEL, new text can be typed to be appended to the previous concatenation before starting a new CAT cycle. For the sake of experimental simplicity, in the experiments described below the CAT optimization in Eq. (5.9) was simulated, rather than actually computed.

### 5.4.5 Corpora and evaluation

The ideas and techniques proposed here have been assessed through some experiments in the framework of the TT2 project [47]. One of the tasks considered in this project is the translation of XEROX technical manuals written in English into Spanish, French and German. Only the translation from English into Spanish is considered here. Consequently, the target spoken language is Spanish.

#### Corpus features

We use three data sets in our experiments. The first one is the standard English-Spanish XEROX benchmark parallel text used in the TT2 project. The training part of this corpus is used here mainly to estimate the translation model parameters needed MT and CAT. The target (Spanish) part of this training set is further used to train most of the language models required in the speech decoding and speech-enabled CAT experiments.

The second data set is the relatively large speech corpus, containing phonetically balanced spoken sentences in Spanish[33]. This corpus was already employed in Chapter 3. It is used here only to train the acoustic HMMs needed in all the speech-related experiments.

The last data set consists of utterances of fragments of target-language (Spanish) sentences, extracted from the test part of the original parallel TT2 corpus. These utterances are used as a test-set to simulate real interactions of the CAT system with human translators.

**Table 5.1:** Features of the text English-Spanish translation XEROX Corpus (K= $\times 1,000$)

| | | English | Spanish |
|---|---|---|---|
| Training-set | Running words | 572K | 657K |
| | Vocabulary | 26K | 30K |
| Test-set | Running words | 7.6K | 9.4K |
| | Running characters | 47K | 59K |
| | Perplexity (3-gram) | 103 | 61 |

All the speech data was acquired using high quality microphones and 16 KHz sampling frequency. A summary of relevant features of these corpora is shown in Tables 5.1, 5.2 and 5.3.

**Table 5.2:** Features of the Spanish speech acoustic training corpus (K= $\times 1,000$)

| | |
|---|---|
| Speakers | 164 |
| Running words (4 hours) | 42K |

**Table 5.3:** Spanish speech test utterances (from the XEROX-corpus)

| | | |
|---|---|---|
| Text | Number of original complete sentences | 128 |
| | Number of different sentence fragments uttered | 485 |
| | Average uttered fragment length and range (words) | 2.4 [1,13] |
| | Average prefix length and range (words) | 4.5 [0,23] |
| | Running words | 1,138 |
| | Running characters | 7,320 |
| Speech | Number of speakers | 10 |
| | Number of utterances | 5,796 |
| | Running words | 13,998 |

The set of test utterances described in Table 5.3 was obtained as follows. First a subset of 128 sentence pairs was selected from the text partition of the XEROX text corpus. For the target (Spanish) sentence of each of these pairs, several segmentations into prefixes and suffixes were randomly performed and, for each generated suffix, a set of prefixes was randomly derived. All the prefixes of suffixes generated in this way constitute the set of sentence fragments uttered by several speakers. In order to approach real CAT user interactions as much as possible, this generation process was performed in such a way that the lengths of the generated fragments were similar to the lengths of accepted parts of system suggestions observed in text-only experiments with a real CAT system applied to the original set of 128 sentence pairs (see next section for details). An example of prefixes/suffixes derived from

the sentence *"adición de fuentes a la lista de recursos"*, along with the resulting fragments produced for speech acquisition, is shown in Table 5.4.

Each of the utterances obtained in this way is the acoustic signal denoted by $\mathbf{x}$ in the previous sections. In the experiments described in the next section, the corresponding text fragment will constitute the reference transcription against which the decoded result, $\hat{\mathbf{d}}$, will be compared. On the other hand, for the experiments related to scenarios DEC-PREF, CAT-PREF and CAT-SEL, the prefix corresponding to each fragment will constitute the *fixed target prefix*, denoted by $\mathbf{e}_p$ in the previous sections. Finally, for the experiments with CAT-SEL the suffix corresponding to each fragment will be used to simulate the suggestion made by the CAT system, denoted as $\hat{\mathbf{e}}_s$ in the previous sections.

**Table 5.4:** Examples of some prefixes, suffixes and prefixes of suffixes randomly derived for the sentence *"adición de fuentes a la lista de recursos"*

| Prefix | Suffix | Prefixes of the suffix |
|---|---|---|
| adición de fuentes | a la lista de recursos | a la lista de |
| adición de fuentes a | la lista de recursos | la, la lista de |
| adición de fuentes a la lista | de recursos | de recursos |
| adición de fuentes a la lista de | recursos | recursos |

### Quality evaluation

Different evaluation measures are needed to assess the quality of speech decoding and CAT operation. Speech decoding accuracy is assessed in terms of conventional measures (where by "*sentences*" we mean the word sequences described above). Specifically the Word Error Rate (WER) and Sentence Error Rate (SER) metrics previously described have been used.

Before presenting the figures employed to assess CAT performance, let us briefly recall the way in which the human translator would interact with this system. When the system provides its best translation suffix suggestion, the human translator will accept a (possibly void) prefix of this suffix. This text fragment is selected by adequately positioning the cursor by means of mouse (or equivalent keyboard) actions. After this point, raw text is typed. Obviously in a laboratory experiment users are not available, but the reference target text can be used to simulate user operation and to measure the following relevant figures to assess the translation/prediction performance of a CAT system:

- *Mouse Action Ratio* (MAR): Number of mouse (or equivalent keyboard) actions needed to position the cursor at the end of the acceptable part of the system suggestion, divided by the total number of running characters. A mouse action is assumed to span at least one word (i.e. 5.4 characters on average).

- *Key Stroke Ratio* (KSR): Number of key strokes plus the final acceptance keystroke needed to type the text necessary to correct the remaining parts of system suggestions in order to produce a target text which exactly matches the reference translation, divided by the number of running characters [13].

- *Accepted Suggestion Fragment Length* (ASFL): Average length of number of words of accepted suggestion fragments.

In addition to these translation error measures, another common (non CAT) translation quality metric is also used:

- *Translation Word Error Rate* (TWER): Minimum number of word substitution, deletion and insertion operations needed to convert a full target sentence provided by a MT system into the corresponding reference translation, divided by the total number of words in the reference translation [1, 7].

### 5.4.6 Experimental results

Different experiments have been carried out to assess the feasibility and potential of the ideas and techniques proposed. Some of the results are text-only experiments aimed at providing performance figures relative to the MT models and CAT system, in the framework of which the speech-related experiments have been performed. The other results are directly devoted to assess the accuracy of speech decoding under increasingly constrained speech-enabled CAT scenarios.

#### MT and CAT text-only experiments

These experiments were conducted by training MT and CAT systems with the training part of the English-Spanish XEROX corpus (Table 5.1) and testing the performance on both the (FULL) test set from Table 5.1 and the (SMALL) subset of 128 selected complete sentences used to generate the sentence-fragments test set of Table 5.3. The MT and CAT systems were based on stochastic finite-state transducers, trained with the GIATI approach [12, 10, 14, 13]. Results are summarized in Table 5.5.

**Table 5.5:** MT and CAT performance on the English-Spanish full test set of Table 5.1 and the small complete-sentences subset of Table 5.3

| | MT | CAT | | |
|---|---|---|---|---|
| | TWER(%) | KSR(%) | MAR(%) | ASFL(Words) |
| FULL | 42.7 | 17.6 | 2.0 | 2.6 |
| SMALL | 57.4 | 24.6 | 3.2 | 2.3 |

As it can be observed, the (randomly selected) SMALL test set turned out to be a particularly difficult subset of the FULL benchmark test set of the English-Spanish

XEROX corpus [12]. According to the KSR and MAR figures, this CAT system could save about 80% of human effort for the FULL test set and 72% for the SMALL one (the joint contribution of KSR and MAR is a rough estimation of the overall translation burden). In contrast with the relatively poor accuracy obtained for MT, these CAT performance figures are quite adequate. Such an improved behavior is clearly achieved thanks to the information extracted from the user feedback in the interaction process.

The low KSR is consistent with the relatively large lengths of the suggested fragments accepted as such (ASFL); on average, in each interaction cycle, the user could accept 2.3 correct words. This ASFL figure was used in section 5.4.5 to guide the generation of target text fragments summarized in Table 5.3, leading to an average generated fragment length of 2.4 words.

These laboratory results correlate reasonably well with real tests with human translators [32], where this CAT system allowed to double human translator productivity in some cases.

Also worth noting in Table 5.5 is the distribution of predicted interaction effort in terms of KSR and MAR. While KSR predicts raw human typing effort (to amend inadequate parts of the system suggestions), MAR accounts for cursor-positioning effort, typically using the mouse. For a skilled typist, these positioning actions tend to be rather "distracting" so that, according to the real human tests, each positioning action tends to be more human-time demanding than raw typing. For instance, in the real tests described in [32], some human translators spent as much as about 15 seconds in many of their cursor-positioning actions (this elapsed time includes reading the system suggestions and deciding the correct follow-up translation/correction).

As a consequence, unless the suggestions are sufficiently good and long, users tend to prefer typing by their own than accepting text suggested by the system. This is exactly the trend we aim to counter with speech-enabled CAT. Using speech for positioning and accepting suggestions, the user would just have to keep reading the suggested text as long as it is acceptable. Clearly, this seems much more natural and less distracting than having to keep switching all the time between raw typing, (mentally) reading the suggestions and positioning the cursor. Of course, speech-driven actions are only expected to be acceptable by users if a sufficiently high recognition accuracy can be achieved, which lead us to the experiments described in the next subsection.

#### Speech-enabled CAT experiments

The aim of these experiments is to assess the feasibility of speech-enabled CAT systems such as those corresponding to the CAT-PREF and CAT-SEL scenarios described in section 5.4.

In order to provide reasonable baselines for these experiments, two additional non CAT-related, speech-decoding experiments were performed in which the source text is not taken into account. These experiments correspond to the settings DEC and DEC-PREF described in section 5.4. DEC corresponds to conventional speech

decoding (of sentence fragments), while in DEC-PREF the search space is more constrained by taking into account the given target sentence prefixes. The experiments will be presented following a least-to-most constrained order; that is, DEC, DEC-PREF, CAT-PREF and CAT-SEL.

Regarding the experiment conditions, they are basically the same as were described in section 3.4. Monophone HMMs trained from MFCCs coefficients. Lexical models represented as finite state automata. Finally, smoothed 3-gram language models were used in all the scenarios (except CAT-SEL, which requires a special language model. See section 5.4).

For the first experiment, DEC, the 3-gram was trained on complete sentences from the (30 Kword vocabulary) Spanish part of the XEROX corpus (Table 5.1), using the SRILM toolkit [48]. Since the test set (Table 5.3) consists of sentence fragments, rather than complete sentences, search was allowed both to start and to end in any 3-gram state.

In the second experiment, DEC-PREF, the information provided by the prefix $\mathbf{e}_p$ is incorporated by considering as initial states in the language model automaton only those which are compatible with this prefix.

In addition to the constraints applied in previous speech-recognition-only experiments, in the third experiment, CAT-PREF, the speech recognizer is further restricted to generate sentences that contain possible translations of words which appear in the source sentence. The required stochastic dictionary was trained using the whole XEROX bilingual training corpus (Table 5.1).

The last experiment corresponds to the most constrained scenario, CAT-SEL. Here the human translator is assumed to just *read aloud* a prefix of the target text suggested by the system. Therefore, in each interaction, a much more constrained LM is built, which only accounts for all the possible prefixes of the system-suggested text.

Results are shown in Table 5.6 [a]. In addition, in Table 5.7, the results corresponding to the different approaches for CAT-PREF scenario (described in section 5.4.3) are also reported.

**Table 5.6:** Speech decoding results (in %) for different scenarios. The average sentence decoding time is also shown

| | DEC | DEC-PREF | NCAT-PREF | CAT-SEL |
|---|---|---|---|---|
| WER | 26.8 | 24.6 | 20.9 | 1.8 |
| SER | 48.3 | 39.0 | 33.9 | 3.7 |
| Av. Time (s) | 29 | 25 | 1.3 | 0.2 |

As expected, speech recognition accuracy increases as the LMs become more constrained. If only prefix-derived constraints are added to DEC, an improvement

[a] Notice that these results do not directly correspond to those presented in [51]. On the one hand, the WER results there presented are actually CER (Character Error Rate) results. On the other, CAT-PREF and DEC-PREF scenarios have been re-implemented by using smoothed $n$-gram models instead of finite state grammars.

**Table 5.7:** Speech decoding results (in %) for different approaches to CAT-PREF scenario. As it is described in section 5.4.3, UCAT-PREF and NCAT-PREF are implemented by re-scoring the unigram part and a whole $n$-gram model respectively with a statistical dictionary. IBMCAT-PREF is based on re-scoring a list of ASR $n$-best with an IBM 1 translation model. Finally IBMLLCAT-PREF is similar to IBMCAT-PREF. The only difference is that a log-linear translation model, including the direct and inverse IBM 1 model, was used here

| | UCAT-PREF | NCAT-PREF | IBMCAT-PREF | IBMLLCAT-PREF |
|---|---|---|---|---|
| WER | 23.2 | 20.9 | 20.1 | 19.8 |
| SER | 34.5 | 33.9 | 32.3 | 31.8 |

of 2.2 points of WER and 9.3 points of SER is obtained in DEC-PREF. By further including constraints derived from the source text, a new improvement is achieved in CAT-PREF: 3.7 points of WER and 5.1 points of SER (or 5.9 points of WER and 14.4 points of SER with respect to the least constrained baseline). In addition to these accuracy improvements, the use of source sentence derived constraints causes a significant decreasing in the system response time, making possible the use of this kind of speech interfaces in real situations. On the other hand, the results for the different approaches to CAT-PREF reported in Table 5.7 do not show a significant difference in performance. The simple NCAT-PREF approach turns out to be very competitive while the use of a whole log-linear translation model (IBMLL-CAT-PREF) translates into a modest improvement of one point in WER and two points in SER.

The constraints added in the last scenario, CAT-SEL, are derived from the (simulated) suggestions of the CAT system. In this case, the improvement is very important: 19.1 points of WER and 30.2 points of SER with respect to the last scenario (or 25 points of WER and 44.6 points of SER with respect to the baseline).

All these results clearly suggest that using knowledge about the source sentence is more important than using only user-validated prefixes. Moreover, if the translation difficulty of the test-set is taken into account (which according to Table 5.5 is quite large), the impact of using the translation information is remarkable. Therefore, better results are expected for CAT-PREF with less problematic texts and/or better (use of) translation models.

The decoding computational demands of the different systems are also worth mentioning. With respect to *memory requirements*, the size (number of 3-grams or edges) of the LMs required by DEC and DEC-PREF are similar, close to 300K. This is about one order of magnitude larger than the size of the LM for CAT-PREF and more than four orders of magnitude larger than the average CAT-SEL LM size. This abrupt complexity drop appears reflected in the accuracy figures shown in Table 5.6. On the other hand, with respect to the *competing time*, a notable reduction is observed from DEC to DEC-PREF and the response time is about twenty times faster from DEC-PREF to CAT-PREF, allowing the implementation of this interface in a real application. Finally CAT-SEL only requires very light computing, which would make it easy to implement this kind of speech-enabled CAT systems on low-end desktop

computers or low-cost dedicated devices. Using the adequate unexpensive hardware, real-time operation can be easily achieved in both CAT scenarios.

The important performance gap observed from CAT-PREF to CAT-SEL is clearly consistent with the much harder constraints of CAT-SEL. As discussed in section 5.3.1, in between these two scenarios, an intermediate setting can be considered, corresponding to Eq. (5.12), which is expected to lead to performance figures falling between those of CAT-PREF and CAT-SEL. However, as this setting requires some speech data (fragments) different from those we have used here, we have left its implementation for future work.

The relative accuracy that can be achieved in the different scenarios has been assessed in terms of WER and SER. However, at least for the CAT-SEL setting, only the SER figure really matters, since it directly estimates the voice-driven cursor-positioning accuracy. The 3.7% SER achieved means that a manual (mouse or keyboard) correction of cursor position is required every 28 voice-driven successful actions, on average. We think this figure can be easily improved by using better acoustic modeling for speech decoding.

As a final remark about the experiments reported in this section, we would like to recall that they can only be considered as simulated scenarios. This is not only because the speech data have not been acquired through real human-CAT interactions, but also because reference target text fragments are used in place of real CAT system suggestions. As discussed along this work, a real setting would involve true human translator interactions and this is not considered adequate at these early stages. Note also, however, that the simulation is very close to the real experiment. In the real case, the user is expected to utter only correct fragments of target text and this is exactly what we do here. Moreover, in order to better approach the real situation, the text fragments to be uttered were generated under the constraint that their average length was as close as possible to that observed in the suggestions made by the system in our text-only CAT experiments (cf. Tables 5.3 and 5.5). Therefore, we think that the obtained results constitute a reasonable starting point in this speech-enabled CAT framework to be further developed in the future.

## 5.5 Adaptive learning

In the different instantiations of IPR addressed in this work, adaptive learning has been considered in one way or another. Although in this chapter we are not concerned about a specific IPR application but about a multi-modal IPR interface, we still have an opportunity to benefit from this of learning approaches.

In the CAT-PREF scenario, Eq (5.17) is maximized to decode the input speech. Here, the three models involved in the maximization are different both structurally and from the point of view of the training data. For that reason, the log-linear model based approximation, previously described in section 5.4.3 can be followed to better benefit from the models involved. This way, instead of using Eq (5.17), a different score function is actually used:

$$\begin{aligned}\hat{\mathbf{d}} &\approx \underset{\mathbf{d}}{\text{argmax}}\, exp\left[\lambda_1 logP(\mathbf{f} \mid \mathbf{e}_p, \mathbf{d}) + \lambda_2 logP(\mathbf{d}\,|\mathbf{e}_p) + \lambda_3 logP(\mathbf{x} \mid \mathbf{d})\right] \\ &= \underset{\mathbf{d}}{\text{argmax}}\, \lambda_1 logP(\mathbf{f}|\mathbf{e}_p, \mathbf{d}) + \lambda_2 logP(\mathbf{d}\,|\mathbf{e}_p) + \lambda_3 logP(\mathbf{x}|\,\mathbf{d}) \qquad (5.20)\end{aligned}$$

Going further, we can include the model defined in Eq. 5.19, which, according to the results, performs better than using the single translation model in Eq (5.17) to achieve the expression in Eq. 5.21:

$$\begin{aligned}\hat{\mathbf{d}} \approx \underset{\mathbf{d}}{\text{argmax}} \quad & \lambda_1 logP(\mathbf{f} \mid \mathbf{e}_p, \mathbf{d}) + \lambda_2 logP(\mathbf{e}_p, \mathbf{d} \mid \mathbf{f}) + \\ & \lambda_3 logP(\mathbf{d}\,|\mathbf{e}_p) + \lambda_4 logP(\mathbf{x} \mid \mathbf{d}) \qquad (5.21)\end{aligned}$$

Notice that this oprimization does not deal with a true probability distribution, since the usual normalization factor required in log-linear models is missing. This factor can be really hard to compute (e.g., the sum over all possible speech signals $\mathbf{x}$is required), but it is not actually necessary since we only try to find a decoding with the best value for the score function, which is the same with or without the normalization factor.

From this, in order to improve the interface accuracy, we can perform an estimation of the values of the $\lambda_i$ parameters based on the sentences already decoded. Because of the application considered, we can expect that, after a speech recognition, the user will correct all the mistakes made by the interface so that the real transcription of the speech input will be available. This information can be used to attempt an estimation of these $\lambda_i$ parameters for each speaker in the test set (we can expect different optimal parameters for different speakers).

After decoding a speech utterance, we have an set of hypotheses scored with both speech and translation models. In addition, we can find out what hypothesis was the correct one according to the user reaction to the recognition result. This way, it would be possible to readjust the $\lambda_i$ parameters in order to improve the score of the correct transcription. The exact procedure consists in firstly computing a set of possible $\lambda_i$ values to posteriorly keep the one making the reference transcription occupy the highest position within the $n$-best list. This way, we have the optimal value for the sentence just recognized (and user validated and/or corrected). Based on the values obtained from all the previous sentences, we can compute the $\lambda_i$ values for the sentence to be decoded as the arithmetic mean of the values already obtained. As can be observed in Table 5.8, this technique achieves an improvement of two points in SER. Notice here, that the number of sentences available for each speaker is about 530 and, therefore, a better estimation could be expected in a real scenario where more interactions are performed by a single user.

**Table 5.8:** Word and Sentence Error Rate using adaptive learning .

| | ADAPTIVE IBMLLCAT-PREF | BASELINE IBMLLCAT-PREF |
|---|---|---|
| **WER,SER** | 19.1, 29.9 | 19.8, 31.8 |

## 5.6 Improving the accuracy of CAT-PREF by imposing simple user constraints

According to the results presented, the accuracy of the CAT-PREF scenario should be improved to be really useful in real tasks. One of the main problems in this scenario is that we are dealing with sentence fragments and, therefore, the language model can not properly model the end of sentence event (every word has the same probability of being an end-of-sentence word). If compare the perplexity of the DEC-PREF "baseline" scenario (71) with this same task but based on whole sentences (40) we can see a significant increment in the task difficulty. As a consequence, a significant amount of errors in this scenario are caused by wrong insertions (as well as deletions) at the end of the partial sentence decoded.

As a possible solution, we can constrain the user inputs, so that the number of words uttered is known in advance. CAT-PREF scenario is aimed at introducing amendments at some CAT hypotheses. In IPR, we expect simple and short corrections since we are trying to minimize the user effort. Hence, it is realistic to assume that we can constrain the utterances in this scenario to have exactly one or two words. Regarding the implementation of this new scenario, a simple solution relies on using the $n$-best list from the ASR in order to filter all the hypotheses that do not have the prefixed length. A better solution is to modify the *Viterbi* search so that all the hypotheses exceeding the length considered in each case are removed from the decoding search space.

As it can be seen in Table 5.9, the performance can be drastically improved by imposing this kind of constraints (in the case of the 1-word constraint, the SER is reduced in 50%).

**Table 5.9:** Word and Sentence Error Rate for two constrained scenarios where the user is required to utter exactly 1 or 2 words

| | WER, SER | |
|---|---|---|
| | DEC-PREF | CAT-PREF |
| 1-WORD | 20.6, 20.6 | 15.1, 15.1 |
| 2-WORD | 19.5, 32.4 | 14.9, 26.1 |

## 5.7 Summary of contributions

In this chapter, we have introduced multi-modality within an IPR system. Specifically, a speech interface for a CAT has been discussed.

After formalizing a general case, several interaction scenarios have been derived. These scenarios represent different theoretical and practical possibilities.

In addition, we have proposed different techniques to take advantage of the environment provided by a CAT system, in order to improve the performance of a speech recognizer. As a result, significant accuracy improvements have been obtained as well as a much better recognition response time. Sufficiently good speech decoding performance has been achieved at least in one scenario, which entails significant potential savings of human effort with respect to typing the whole target text or having to use non-speech cursor-positioning actions. Good performance (both speed and accuracy) is achieved thanks to search constraints derived from both the source sentence (through a translation model) and the successively consolidated prefixes of the target text.

A simple adaptive learning procedure has been described in order to adapt the speech interface to a specific user without requiring any kind of extra user effort.

Finally a simple method to constrain the kind of inputs allowed has been proposed aimed at significantly improving the interface performance.

CHAPTER 6

# CONCLUSIONS

In this final chapter, a summary of the main work developed will be presented as well as some ideas to continue the different lines proposed.

## 6.1 Main contributions

The main goal of this work has been to explore the application of the Interactive Pattern Recognition (IPR) paradigm to several natural language applications.

In the first place, a formalization of IPR has been proposed in order to establish a general framework not only for the applications being developed but also for the new lines of work that can be followed in the future. To this end, the user activity has been included into the statistical formulation of the pattern recognition approach. In the case of NLP a first order process has been considered to model the user actions. The point is that the last user action can reflect the whole history of the process, which can be seen as reasonable taking into account the sequential nature of the human language. As a result, a general scenario can be stated for all the applications that can be derived from this.

In addition, some interesting issues, heavily related to IPR, have been discussed. Multi-modality, on the one hand, as an interesting way to deal with the user actions. Multi-modality does not arise in IPR by accident. On the contrary, the IPR nature itself intrinsecally entails and promotes the inclusion of multi-modality. On the other hand, IPR constitutes a good opportunity to apply adaptive learning approaches. The system is always being supervised during its normal operation mode and we can benefit from the outcomes produced to improve the models involved in the recognition process. Simple approximations to including adaptive learning have been proposed for the different applications considered and, in all the cases, significant improvements have been achieved.

The first IPR application proposed (CAST) allows to generate perfect speech transcription by developing an interactive system that cooperates with a human transcriber. The system is able to provide transcriptions of an input speech fragment and, at the same time, to adapt the results obtained to the user feedback. This way,

we can reduce the effort required to perform the transcription. Several experiments have been presented in order to support the CAST approach. The results obtained show that a significant amount of user effort can be saved following this proposal. Nevertheless, the initial approximation to CAST proved to be inefficient (and possibly useless for the time being), due to the excessive system response time. To solve the problem, an alternative based on the use of word graphs has been considered. At the expense of a minimum performance degradation, this alternative has proven to be able to dramatically decrease the system latency, making CAST usable as an interactive tool.

Taking the IPR approach to an extreme environment, where no input pattern is really available and only the user feedback is present, a new application (Interactive Text Generation, ITG) has been proposed. The point here is to provide assistance in situations where typing becomes difficult. In the first place, different experiments have been carried out to confirm the optimality of a greedy approach to this problem when compared with the traditional maximization of the posterior probability by means of a dynamic programming algorithm. Next, a new prediction modality, based on responding to each user key stroke was addressed. Different models were proposed to deal with this situation.

In addition adaptive learning was thoroughly studied for this application. Some alternatives were proposed and we also attempted to study the behavior of adaptive learning from the user point of view. Finally, two applications of the ITG approach, apart from merely generating text documents were discussed. It is worth to emphasize the possibilities of using ITG in information retrieval and the initial and promising results achieved.

Finally, multi-modality in IPR was addressed. Specifically, the case of a multimodal speech and text interface for Computer Assisted Translation (CAT) was discussed. To this end, a formalization of the inclusion of speech in CAT was developed and different scenarios for this multi-modal interface were considered.

## 6.2 Selected derived publications

A selection of the publications derived from this work are briefly commented.

The general IPR framework (Chapter 2) is described in:

> Enrique Vidal, Luis Rodríguez, Francisco Casacuberta, Ismael García Varea. *Computer Assisted Pattern Recognition*. 4th Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms. Brno (Czech Republic) LNCS.

Regarding CAST (Chapter 3), the initial proposal along with several experiments can be found in:

Luis Rodríguez, Francisco Casacuberta, Enrique Vidal. *Computer Assisted Transcription Of Speech*. III Iberian Conference on Pattern Recognition and Image Analysis. Girona (Spain). LNCS.

A directly derived application of the CAST proposal for Interactively transcribing handwritten documents is described in:

Verónica Romero, Alejandro H. Toselli, Luis Rodríguez, Enrique Vidal. *Computer Assisted Transcription for Ancient Text Images*. International Conference on Image Analysis and Recognition. Montreal (Canada) LNCS.

Verónica Romero, Alejandro H. Toselli, Enrique Vidal, Luis Rodríguez. *Computer Assisted Transcription of Handwritten Text Images*.9th International Conference on Document Analysis and Recognition. Curitiba (Brazil)

A description of the application of ITG to information retrieval (Chapter 4) can be found in:

Luis Rodríguez, Alejandro Revuelta, Ismael García-Varea, Enrique Vidal. *Interactive text generation for information retrieval*. 10th International Workshop on Pattern Recognition in Information Systems. Pending to be published

Finally, the inclusion of multi-modality into a CAT system (Chapter 5) is discussed in the following publications:

Enrique Vidal, Francisco Casacuberta, Luis Rodríguez, Jorge Civera, Carlos Martínez. Computer Assisted Translation Using Speech Recognition. IEEE Transactions on Audio, Speech and Language Processing. 2006. 14(3):941-951.

Luis Rodríguez, Enrique Vidal, Francisco Casacuberta, Jorge Civera, Carlos Martínez. On the use of speech recognition in computer assisted translation. Interspeech 2005. Lisbon (portugal)

In addition, in Chapter 5 , the problem of aligning a source sentence and a fragment of a target sentence was posed. To solve this problem, a series of algorithms were proposed in:

Luis Rodríguez, Ismael García Varea, José Antonio Gámez. On the application of different evolutionary algorithms to the alignment problem in statistical machine translation Neurocomputing. Volume 71 , Issue 4-6 (January 2008)

## 6.3 Future work

This work is only a starting point in the exploration of the IPR paradigm and the applications that can be derived from it. There is a great amount of open fields to be explored. We will only give some examples of possible general lines of work to be followed.

- The IPR framework can be extended and adapted to other kind of pattern recognition tasks. For instance, multimedia information recovering seems to be a really good scenario to develop IPR.
- In the case of CAST, there are two general lines that can be proposed. On the one hand, the inclusion of a multi-modal speech interface, as was developed for CAT, can be adequate since speech recognition is included into the application itself. On the other hand, the combination of CAST+CAT can be more than justified in order to address the problem of high quality speech translation.
- Regarding ITG, the use of long-dependency language models should be addressed to improve the prediction accuracy. Some experiments were developed in this sense, but the modest improvements achieved did not justify the inclusion of such results in this thesis. In addition, the use of ITG as a human learning tool is worth to be mentioned. In the case of the two application proposed (computer programming and information retrieval) experiments involving real users are needed to prove the usefulness of this proposal.

# BIBLIOGRAPHY


[1] J.C. Amengual, J.M. Benedí, F. Casacuberta, A. Castaño, A. Castellanos, V. Jiménez, D. Llorens, A. Marzal, M. Pastor, F. Prat, E. Vidal, and J.M. Vilar. The EuTrans-I Speech Translation System. *Machine Translation*, 15:75–103, 2000.

[2] S. Bickel, P. Haider, and T. Scheffer. Predicting sentences using n-gram language models. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 193–200, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

[3] J. Brousseau, C. Drouin, G. Foster, P. Isabelle, R. Kuhn, Y. Normandin, and P. Plamondon. French speech recognition in an automatic dictation system for translators: the TransTalk project. In *Proceedings of the Forth European Conference on Speech Communication and Technology (Eurospeech 95)*, pages 193–196, Madrid, Spain, 1995.

[4] P. Brown, S. Chen, S. Della Pietra, V. Della Pietra, S. Kehler, and R. Mercer. Automatic speech recognition in machine aided translation. *Computer Speech and Language*, 8:177–187, 1994.

[5] P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roosin. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85, 1990.

[6] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

[7] F. Casacuberta, H. Ney, F. J. Och, E. Vidal, J. M. Vilar, S. Barrachina, I. García-Varea, D. Llorens, C. Martínez, S. Molau, F. Nevado, M. Pastor, D. Picó, A. Sanchis, and C. Tillmann. Some approaches to statistical and finite-state speech-to-speech translation. *Computer Speech and Language*, 18:25–47, 2004.

[8] F. Casacuberta, E. Vidal, A. Sanchis, and J. M. Vilar. Pattern recognition approaches for speech-to-speech translation. *Cybernetic and Systems: an International Journal*, 35(1):3–17, 2004.

[9] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language. Technical report, 1996.

[10] J. Civera, J. M. Vilar, E. Cubel, A. L. Lagarda, S. Barrachina, E. Vidal, F. Casacuberta, D. Picó, and J. González. From machine translation to computer assisted translation using finite-state models. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP04)*, pages 349–356, Barcelona, Spain, 2004.

[11] 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.

[12] J. Civera, J.M. Vilar, E. Cubel, A.L. Lagarda, F. Casacuberta, E. Vidal, D. Picó, and J. González. A syntactic pattern recognition approach to computer assisted translation. In A. Fred, T. Caelli, A. Campilho, R. P.W. Duin, and D. de Ridder, editors, *Advances in Statistical, Structural and Syntactical Pattern Recognition – Joint IAPR Imternational workshops on Syntactical and Structural Pattern Recognition (SSPR 2004) and Statistical Pattern Recognition (SPR 2004)*, Lecture Notes in Computer Science. Springer-Verlag, Lisbon, Portugal, August 2004.

[13] E. Cubel, J. Civera, J. M. Vilar, A. L. Lagarda, S. Barrachina, E. Vidal, F. Casacuberta, D. Picó, J. González, and L. Rodríguez. Finite-state models for computer assisted translation. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI04)*, pages 586–590, Valencia, Spain, 2004.

[14] E. Cubel, J. González, A. Lagarda, F. Casacuberta, A. Juan, and E. Vidal. Adapting finite-state translation to the TransType2 project. In *Proceedings of the Joint Conference combining the 8th International Workshop of the European Association for Machine Translation and the 4th Workshop on Controlled Language Applications (EAMT/CLAW 03)*, Dublin, Ireland, 2003.

[15] J.E. Díaz-Verdejo, A.M. Peinado, A.J. Rubio, E. Segarra, N. Prieto, and F. Casacuberta. Albayzin: a task oriented spanish speech corpus. In *Proceedings of First Intern. Conf. on Language Resources and Evaluation (LREC-98)*, volume 1, pages 497–501, 1998.

[16] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, New York, NY, 2nd edition, 2000.

[17] M. Dymetman, J. Brousseau, G. Foster, P. Isabelle, Y. Normandin, and P. Plamondon. Towards an automatic dictation system for translators: the TransTalk project. In *Proceedings of International Conference on Spoken Language Processing (ICSLP94)*, pages 193–196, Yokohama, Japan, 1994.

[18] E.Vidal. Finite-State Speech-to-Speech Translation. In *Int. Conf. on Acoustics Speech and Signal Processing (ICASSP-97), proc., Vol.1*, pages 111–114, Munich, 1997.

[19] N. Garay-Vitoria and J. Abascal. Text prediction systems: a survey. *Universal Access in the Information Society*, 4(3):188–203, March 2006.

[20] J.C.Amengual and E.Vidal. Efficient Error-Corecting Viterbi Parsing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.PAMI-20, No.10:1109–1116, October 1998.

[21] F. Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, Massachusetts, USA, 1998.

[22] S. Khadivi. *Statistical Computer-Assisted Translation*. PhD thesis, RWTH Aachen University, Aachen, Germany, July 2008.

[23] S. Khadivi and H. Ney. Integration of automatic speech recognition and machine translation in computer-assisted translation. *IEEE Transactions on Audio, Speech and Language Processing*, 16(8):1551–1564, November 2008.

[24] K. Knight. Decoding complexity in word-replacement translation models. *Comput. Linguist.*, 25(4):607–615, 1999.

[25] P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the MT-Summit*.

[26] P. Koehn, H.Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computational Linguistics*, June 2007.

[27] P. Koehn, F. J. Och, and D. Marcu. Statistical phrase-based translation. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL)*, pages 48–54, Edmonton, Canada, May 2003.

[28] R. Kuhn and R. De Mori. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:570–583, 1990.

[29] P. Langlais, G. Foster, and G. Lapalme. TransType: a computer-aided translation typing system. In *Proceedings of the Workshop on Embedded Machine Translation Systems (NAACL/ANLP2000)*, pages 46–52, Seattle, Washington, USA, May 2000.

[30] D. Llorens, F. Casacuberta, E. Segarra, J.A. Sánchez, and P. Aibar. Acoustical and syntactical modeling in ATROS system. In *Proceedings of International Conference on Acoustic, Speech and Signal Processing (ICASSP99)*, pages 641–644, Phoenix, Arizona, USA, March 1999.

[31] I. Scott Mackenzie. A character-level error analysis technique for evaluating text entry methods. In *Proceedings of the Second Nordic Conference on Human Computer Interaction - NordiCHI 2002*, pages 241–244. ACM, 2002.

[32] E. Macklovitch, N. T. Nguyen, and R. Silva. TT2. TransType2. User evaluation report 3, 2004. Information Society Technologies (IST) Programme, IST-2001-32091.

[33] A. Moreno, D. Poch, A. Bonafonte, E. Lleida, J. Llisterri, J. B. Mariño, and C. Nadeu. Albayzin speech database: design of the phonetics corpus. In *Proceedings of the Third European Conference on Speech Communication and Technology (Eurospeech 93)*, pages 175–178, Berlin, Germany, September 1993.

[34] H. Ney, S. Niessen, F. J. Och, H. Sawaf, C. Tillmann, and S. Vogel. Algorithms for statistical translation of spoken language. *IEEE Transactions on Speech and Audio Processing*, 8(1):24–36, 2000.

[35] H. Ney and S. Ortmanns. Extensions to the word graph method for large vocabulary continuous speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 1791–1794, Munich, Germany, April 1997.

[36] Y. Normandin. *Hidden Markov models, maximum mutual information estimation, and the speech recognition problem*. PhD thesis, Montreal, Que., Canada, Canada, 1991.

[37] H. Nyquist. Certain topics in telegraph transmission theory. *Proceedings of the IEEE*, 90(2):280–305, 2002.

[38] F. J. Och and H. Ney. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL 2000).*, pages 440–447, Hongkong, China, October 2000.

[39] F.J. Och, R. Zens, and H. Ney. Efficient search for interactive statistical machine translation. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL03)*, pages 387.–393, Budapest, Hungary, April 2003.

[40] J. Oncina. Optimum algorithm to minimize human interactions in sequential computer assisted pattern recognition. *Pattern Recognition Letters*, 30(6):558–563, February 2009.

[41] S. Ortmanns, H. Ney, and X. Aubert. A word graph algorithm for large vocabulary continuous speech recognition. *Computer Speech and Language*, 11(1):43–72, January 1997.

[42] D. B. Paul and J. M. Baker. The design for the wall street journal-based csr corpus. In *HLT '91: Proceedings of the workshop on Speech and Natural Language*, pages 357–362, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[43] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recoginition. *Proceedings of the IEEE*, 77:257–286, 1989.

[44] A. Reddy, R. Rose, H. Safadi, S. Larkin, and G. Boulianne. Incorporating knowledge of source language text in a system for dictation of document translations. In *Proceedings of the twelfth Machine Translation Summit*. Association for Computational Linguistics, 2009.

[45] A. Revuelta-Martínez, L. Rodríguez, and I. García-Varea. Multilingual access to online help systems and databases. *Procesamiento del Lenguaje Natural*, 44, 2010.

[46] D. Sankoff and J. B. Kruskal. *Time warps, string edits, and macromolecules: The theory and practice of sequence comparison*. Addison Wesley, 1983.

[47] SchlumbergerSema S.A. and Instituto Tecnológico de Informática and Rheinisch Westfälische Technische Hochschule Aachen Lehrstul für Informatik VI and RALI - University of Montreal and Celer Soluciones and Société Gamma and Xerox Research Centre Europe. TT2. TransType2 - computer assisted translation. Project technical annex., 2001. Information Society Technologies (IST) Programme, IST-2001-32091.

[48] A. Stolcke. SRILM - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP02)*, pages 901–904, Denver, Colorado, USA, September 2002.

[49] Christoph Tillmann and Hermann Ney. Statistical language modeling and word triggers. In *In SPECOM?96*, pages 22–27, 1996.

[50] H. Trost, J. Matiasek, and M. Baroni. The language component of the fasty text prediction system. *Applied Artificial Intelligence*, 19(8):743–781, 2005.

[51] E. Vidal, F. Casacuberta, L. Rodríguez, J. Civera, and C. Martínez. Computer-assisted translation using speech recognition. *IEEE Transaction on Audio, Speech and Language Processing*, 14(3):941–951, 2006.

[52] 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.

[53] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. Carrasco. Probabilistic finite-state machines - part II. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1025–1039, 2005.

[54] A. Viterbi. Error bounds for convolutional codes and an asymtotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.

[55] S.J. Young. The htk hidden markov model toolkit: Design and philosophy. *Entropic Cambridge Research Laboratory, Ltd*, 2:2–44, 1994.

APPENDIX A

# APPENDIX. PROTOTYPES

From the IPR applications described in Chapters 3, 4 and 5 a set of prototypes have been developed. The main goal is to show that the theoretical approaches and laboratory experiments can become into a real application that allows a user to experiment the different IPR proposals.

## A.1 Computer assisted speech transcription prototype

The CAST prototype was developed to strength the CAST approach by constructing a user-testable application that reflects and allows a realistic testing of the operation mode of this application. In addition, new approaches related to different interaction aspects could be obtained from the feedback coming from the users testing the prototype.

### A.1.1 Objectives

The main goals of this prototype can be summarized as:

- Constructing a full functional and user-testable version of a tool implementing the CATS approach
- Performing usability test for this tool.
- Including some new ideas about functionality that can improve the system usability and performance. This proposals will be obtained from the usability tests previously mentioned.
- Extending the initial CAST theoretical framework based on the experience achieved from this prototype

### A.1.2 Functional Requirements

Since the main task to be solved by the prototype is the achievement of high quality transcription, by means of a cooperation process between a human user and an automatic speech transcription system, we can define the following functional requirements.

- The system has to be able to provide transcriptions from files containing spoken utterances. This can be accomplished by including an automatic speech recognizer into the prototype.
- The user can interact with the system and this interaction has to be sent to a speech recognizer in order to use this information for future predictions.
- The user has to be able to define work sessions where parameters as: files to be transcribed, files already transcribed, configuration parameters, etc. are saved.
- The prototype must be able to collect different statistics about the CAST process that will be used to improve the prototype itself as well as the CAST theoretical proposal.

### A.1.3 Architecture

The prototype is composed of three different subsystems:

#### Prediction Engine

The prediction engine is a general-purpose, speaker-independent, continuous speech recognizer (ATROS). This engine is a stand-alone application capable of recognizing speech in real time from a microphone input or from a wavefile.

Given an input utterance, the recognizer initially pre-process the signal by performing a border detection algorithm and a pre-emphasis filter to increase the magnitude of the high frecuencies present in the speech signal. Next, a feature extraction is performed to obtain a MFCC (Mel Frequency Cepstral Coefficients) vector for each frame in the input signal. Then, the energy along with the first and second derivatives are added to build the final feature vector for the current frame.

Once the feature extraction has been performed, the recognition process is carried out. The recognizer uses three-state, left-to-right Hidden Markov Models as acoustic models. Finite-state automatons are used as Lexical entries and, finally, language models are implemented as finite-state networks. The Viterbi algorithm is used to find the most probable path on the integrated network (acoustic+lexical+language models). Finally, the utterance represented by this optimal path is returned as the result of the recognition.

#### Graphical User Interface

This interface provides all the interaction mechanisms to allow an efficient communication between the user and the prototype. The interface has been implemented in Java.

Initially, the interface request the user to create a new transcription project or to use a previously created one. A project is a set of input wavefiles to be transcribed. Along with these files a project stores the transcriptions already obtained.

The main screen shows, on the one hand, the list of files in the current project so that the user can choose the file to be transcribed currently. On the other hand, the waveform of the current file is shown in a graphic panel and a text field is used to show the current system suggestion along with the different user interactions performed as it is shown in Figure A.2. The user can interact with this text field by using the mouse (or the keyboard) to select the current prefix. In addition, the keyboard is used to amend the system suggestions.

#### Communication module

Since the prediction engine and the user interface have been developed in different languages, a communication module is needed to send the user feedback from the interface to the engine and to collect the different predictions.

This module is currently implemented by adding a communication API to the recognizer. This API is implemented by using the Java Native Interface (JNI) that allows a java program to call C/C++ functions. This way, this API is included into the recognizer building process so that the interface can make the proper calls in runtime. Three main functions are defined in the API, used for initializing the recognizer, setting a new user prefix and requesting a new prediction from the engine.

## A.2 Interactive text generation prototype

The ITG prototype is intended to test the theoretical proposal with real users and to explore different possibilities for a collaborative user-computer environment in the generation of different text documents, queries, etc.

### A.2.1 Objectives

The main goals of this prototype can be summarized as:

- Constructing a full functional and user-testable version of a tool implementing the ITG approach
- Performing usability tests for this tool.
- Including some proposals about functionality that can improve the system usability and performance. This proposals will be obtained from the usability test previously mentioned.


**Figure A.1:** CAST prototype

- Explore new scenarios for ITP (constrained interfaces, generation of documents in a foreign language, etc.)
- Extending the initial ITP theoretical framework based on the experience achieved from this prototype

### A.2.2 Functional requirements

Since the main task to be solved by the prototype is the semi-automatic generation of text documents, we can define the following functional requirements.

- The system has to be able to provide completions to the user typed text. This can be accomplished by including an automatic text prediction engine into the prototype.
- The user can interact with the system and this interaction has to be sent to the text prediction system in order to include this information for future predictions
- The user has to be able to define work sessions where parameters as files to be generated, documents already transcribed, etc. are defined.
- The prototype must be able to collect different statistics about the ITP process that will be used to improve the prototype itself as well as the ITP theoretical proposal.

## A.3 Architecture

The prototype is composed of three different subsystems:

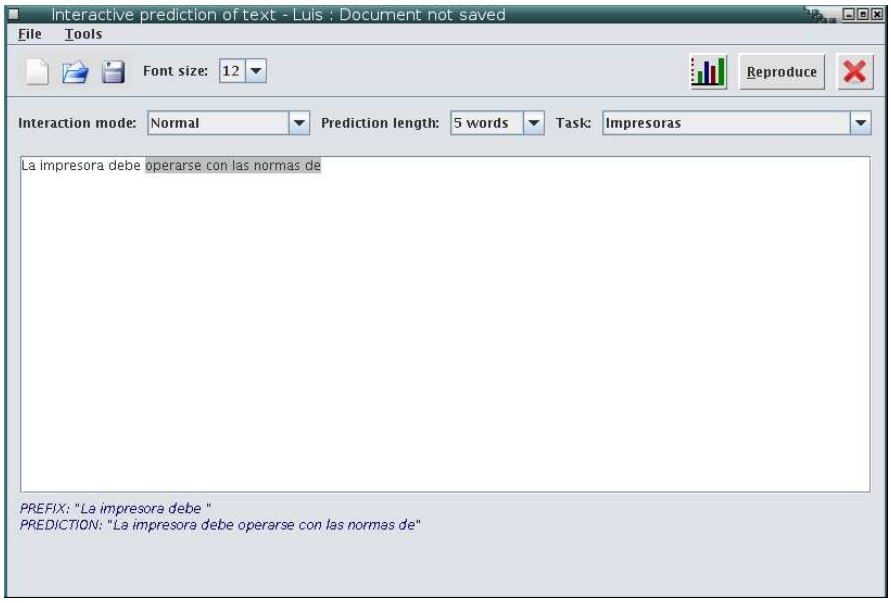



**Figure A.2:** ITG Initial prototype

- An automatic text predictor. Based on $n$-gram language models and implemented in C++. It is able to perform adaptive learning on the test being processed.
- A Graphical User Interface. This interface provides all the interaction engine to allow an efficient communication between the user and the prototype. The interface has been implemented in Java.
- A communication module between the text predictor and the graphical interface that sends the information derived from the user interaction to the predictor. This module is implemented by following a server-client approach and it is based on the use of network sockets.

## A.4 Multi-modal Computer assisted translation prototype

A multi-modal Computer assisted translation (CAT) prototype was built to include a speech and keyboard interface into a CAT application. This prototype is not aimed at testing the CAT approach since other prototypes have been developed to this end but, rather, at experimenting with a multi-modal interface.

### A.4.1 Objectives

The main goals of this prototype can be summarized as:

- Constructing a completely user-testable of a multi-modal CAT interface.
- Performing usability tests for this interface
- Extending the initial multi-modal CAT theoretical scenarios proposed from the results obtained with the use of this prototype

### A.4.2 Functional Requirements

The following functional requirements were established for this prototype:

- The system has to be able to provide initial translations for a source sentence and interactively complete user validated translation prefixes.
- The user can interact with the system by using two different modalities: mouse+keyboard and speech.
- The use of the speech is planned according to the scenarios CAT-PREF and CAT-SEL proposed in Chapter 5.

### A.4.3 Architecture

The prototype is composed of four different subsystems:

#### Speech recognizer

The prediction engine is the general-purpose, speaker-independent, continuous speech recognizer (ATROS) already described in section A.1.3.

#### Translation engine

The translation engine used in the CAT process consisted in a "C" application able to deal with translation models in the form of finite state transducers. The different translation suffixes are obtained by performing an adapted *Viterbi* search on the transducer.

#### Graphical User Interface

This interface provides all the interaction mechanism between the user and the prototype.

Initially, the user can select a list of files containing source sentences to be translated. Next the user can proceed by selecting a specific sentence and the interactive process start. Initially, the system provides an initial full translation for the source sentence. Then, the user can validate part of the translation proposed and introduce some amendments. Taking into account this user feedback the system will provide a new translation suffix.

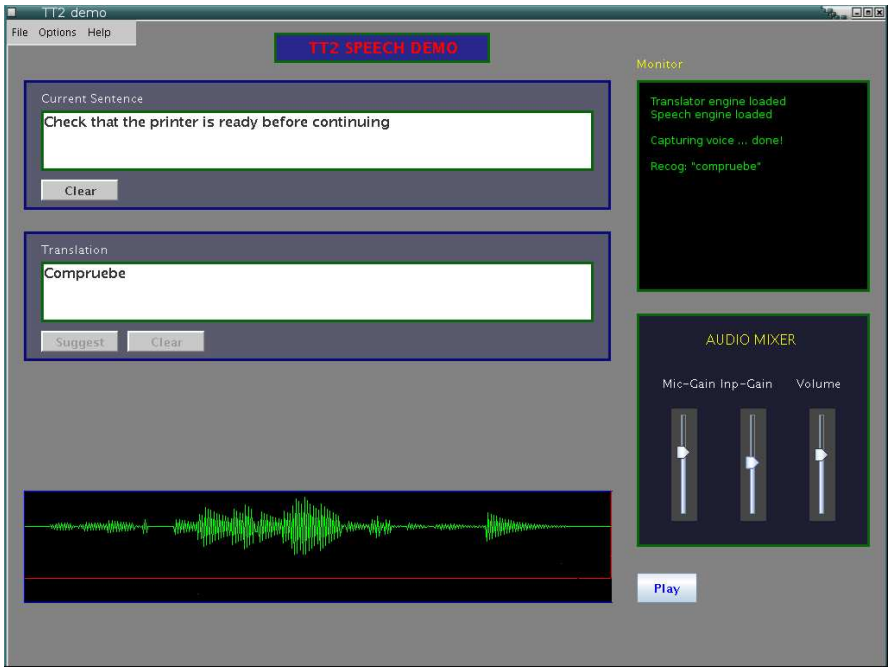



**Figure A.3:** Multi-modal CAT prototype

The user feedback can be performed by either keyboard or speech. In the case of the speech, two different scenarios are considered. When a new system translation suggestion is available, an error-free prefix can be selected by uttering the final word (or words) in this prefix. This corresponds to the CAT-SEL scenario. Once the prefix has been set, the user can dictate a suitable continuation for this prefix (CAT-PREF scenario) and after transcribing the user utterance, the translation prediction engine will suggest a new suffix. This way, the whole CAT process can be performed by using the voice only. Alternatively, the user can employ the keyboard and/or the mouse to correct the speech recognition mistakes or to simply interact with the CAT system in the traditional way.

The main interface screen shows, on the one hand, the source sentence to be currently translated and, on the other, the CAT interactive process. In addition the wave file resulting of a user utterance is also shown. Finally some additional information about the prototype is also presented. In Figure A.3 a snapshot of the application is shown.

### Communication module

The user interface has been developed in Java and the speech recognizer and the translation engine were both developed in C. As a result, the same communication method described in section A.1.3 and based on the Java Native Interface was employed.

# LIST OF FIGURES

# LIST OF TABLES