BUILDING A GRAPHEME-TO-PHONEME CONVERTER

Supervisors:
Antonio Bonafonte
Universitat Politècnica de Catalunya
Tecnologies i Aplicacions del Llenguatge i la Parla
Department of Signal Theory and Communications
Campus Nord
Jordi Girona 1-3 Building D5
08034 Barcelona
Spain

Gerrit Bloothooft
Utrecht University
Utrecht Institute of Linguistics OTS
Institute of Foreign Languages (Linguistics)
ICT-Center Faculty of Arts
Trans 10, 3512 JK Utrecht
The Netherlands

Student:
Albert Roeleveld
Liberal Arts
Specialization: Phonetics
Student number: 9860347

Credits: 16 points old style on C-level (23 points ECTS)
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table of Contents</td>
<td>2</td>
</tr>
<tr>
<td>1 Preface</td>
<td>3</td>
</tr>
<tr>
<td>2 Evaluation</td>
<td>4</td>
</tr>
<tr>
<td>2.1 How did I get my trainee post in Barcelona?</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Information about the UPC and Talp</td>
<td>5</td>
</tr>
<tr>
<td>2.3 Description of practical-assignment</td>
<td>7</td>
</tr>
<tr>
<td>2.4 Description and justification</td>
<td>9</td>
</tr>
<tr>
<td>2.5 Problems</td>
<td>10</td>
</tr>
<tr>
<td>2.6 Are the goals achieved?</td>
<td>11</td>
</tr>
<tr>
<td>2.7 Retrospective</td>
<td>12</td>
</tr>
<tr>
<td>3 Product</td>
<td>13</td>
</tr>
<tr>
<td>3.1 Report: State-of-the-art in Grapheme-to-Phoneme Conversion (GPC)</td>
<td>13</td>
</tr>
<tr>
<td>3.1.1 What is GPC?</td>
<td>13</td>
</tr>
<tr>
<td>3.1.2 Different techniques for handling out of vocabulary words</td>
<td>14</td>
</tr>
<tr>
<td>3.1.2.1 Rules</td>
<td>14</td>
</tr>
<tr>
<td>3.1.2.2 Data-driven</td>
<td>14</td>
</tr>
<tr>
<td>3.1.2.2.1 Decision trees</td>
<td>14</td>
</tr>
<tr>
<td>3.1.2.2.2 Neural networks</td>
<td>15</td>
</tr>
<tr>
<td>3.1.2.2.3 Nearest Neighbor</td>
<td>15</td>
</tr>
<tr>
<td>3.1.2.2.4 Pronunciation by Analogy (PbA)</td>
<td>15</td>
</tr>
<tr>
<td>3.1.2.2.5 Entropy models</td>
<td>16</td>
</tr>
<tr>
<td>3.1.3 Accuracy</td>
<td>17</td>
</tr>
<tr>
<td>3.1.4 Objectives for the near future</td>
<td>17</td>
</tr>
<tr>
<td>3.2 Description of the Pronunciation by Latent Analogy model</td>
<td>19</td>
</tr>
<tr>
<td>3.2.1 Pronunciation by Analogy</td>
<td>19</td>
</tr>
<tr>
<td>3.2.2 Pronunciation by Latent Analogy</td>
<td>20</td>
</tr>
<tr>
<td>3.2.2.1 Orthographic anchors</td>
<td>21</td>
</tr>
<tr>
<td>3.3 The final program</td>
<td>23</td>
</tr>
<tr>
<td>3.3.1 Results</td>
<td>24</td>
</tr>
<tr>
<td>3.3.2 Improvements</td>
<td>24</td>
</tr>
<tr>
<td>4 Literature</td>
<td>26</td>
</tr>
</tbody>
</table>
1 Preface

This report describes the research I have conducted in Barcelona at the Talp research institute. I have worked for four months at Talp in the field of phoneme to grapheme conversion. This is a two-piece report. In the first part I evaluate my practical and my stay in Barcelona. This is a requirement from my home institution to obtain the credits. In the second part I describe the actual ‘product’.

Here is also a proper place to thank some institutions and people. First I would like to thank both universities and the research institute Talp. Utrecht University has given me the ability to go abroad and have given me some financial support in the form of an Erasmus scholarship. The UPC and Talp offered me a place to work and give me time to conduct a research.

My special appreciation goes to Gerrit Bloothoofdt and Antonio Bonafonte, they are my supervisors who have arranged the final agreement for this practical. They were the persons who helped me when I was stuck or when I just need some help.

Albert Roeleveld
2 Evaluation

In the first part of this report I will evaluate my traineeship. This is not only done because it is necessary to obtain the credits, but it is also a good way to look back at this period. First I will describe how I have managed to get my trainee post and give some information about the structure of the UPC (Universitat Politècnica de Catalunya). Secondly I will describe the objectives of this practical. And finally I will look back at my stay at the UPC and review the facts if have learned in the past five months.

2.1 How did I get my trainee post in Barcelona?

During the study Liberal Arts it is obligatory to complete a traineeship. In my specialization phonetics there is space in the curriculum during the last year. From September till January I have the possibility to fulfill this requirement. It was always my intention to complete this part of my study in a foreign country. The contacts with a different culture in a foreign country should make me more experienced in several ways. I will not only discover a new live style but will also see another academic system.

It is in the field of phonetics always difficult to find an interesting trainee post. It is certainly difficult with the recent economical decline. Because of this decline most companies are not interested in trainees. These need supervision and this is a time and therefore a costly financial affair. With this problem bearing in mind it was not an easy task to find a trainee post in a foreign country. Someone told me at a certain moment that it is maybe easier to find a trainee post somewhere at a foreign university or in a closely related research institute. Normally it is possible to participate in a research at a university institute. Utrecht University has excellent contacts with other universities in the world. And with the frontiers disappearing it would be certainly easier to find a trainee post at a research institute somewhere in Europe.

I told this to Gerrit Bloothooft a teacher in linguistics at Utrecht University. His first reaction was that he had many contacts with universities abroad. He asked me in which country I was especially interested to complete my traineeship. My first reaction was somewhere in the south of Europe, not only because of the better weather conditions during the autumn/winter but also because of the relaxed atmosphere. My second reaction was Spain because I know the basics of the Spanish language. And during my stay in Spain I will have a great opportunity to improve this knowledge. Gerrit Bloothooft was so kindly to send some emails to the UPC. The UPC reacted with a positive mail and so the first contacts were there.

The technical university in person of Antonio Bonafonte was well willing to offer me a place at the department of Signal Theory and Communication in the Speech Processing group, which is strongly related with research institute Talp (Tecnologies i Aplicacions del Llenguatge i la Parla). Here I could conduct a research in the field of text-to-speech systems. During the past summer at the 4th Master School on Language and Speech, which was held at the UPC, I met Antonio Bonafonte. Here we agreed on the subject and the agreements were signed. I could start in September.
### 2.2 Information about the UPC and Talp

The UPC is a public institute of higher education whose priority objectives are study, teaching, research and technology transfer. The university is specialized in the fields of architecture, engineering, merchant seaman-ship, economics, health sciences and applied mathematics. The university in the present state is founded in March 1971. Nowadays there are more than 29,000 students and over 2,000 teachers.

The research carried out at the UPC leads the field in many areas and is closely linked to the interest of society, and especially to the production sectors. The research is most of the time state-of-the art and is in closely cooperation with companies. In this way is it possible for the university to anticipate at needs and questions from the business community. The university has not only tight connections with the business community but also with other research centers. Not only in Spain or Europe but also in Latin America. Due to this is it possible for the UPC to stay well informed in the latest developments in many research areas. This all makes the UPC an excellent place to conduct research and a fine place to fulfill my traineeship.

The UPC has many departments, which are not only situated in Barcelona but also in the region of Catalonia. Making the university one of the leading suppliers of employment to high-educated people in the region of Catalonia. I am situated at the Department of Signal Theory and Communication. It is located in the city of Barcelona. This department is subdivided in six groups each with its own research field. The Signal Processing Group is the part where there is research in language and speech. To be more precise this practical is conducted at the subgroup Speech Processing and within Talp Research Center. (See gray shaded box in the organization chart above.) The people belonging to this group are working in the field of speech recognition and text-to-speech conversion.

Talp is a research center for technologies and applications in language and speech. It is a specific research center of the UPC entirely committed to technology and applications of natural language processing techniques, either for spoken or written language. There are around 40 researchers active in this center; most of them are lecturers either in the Telecommunication or Informatics curricula at the UPC. The researchers are divided in two groups.
One research group is the Natural Language Processing Group. They are working in the field of automatic written language processing techniques and applications. The main working areas are the multilingual lexical resources generation and use, the documents information extraction, natural language interfaces design and language treatment basic techniques design (morpho-syntactic disambiguation, entities recognition, semantic representation, etc.). The group working languages are Spanish, Catalan and English.

The other research group from Talp is the Speech Processing Group. This group has published in the last five years more than hundred basic investigation articles. The working areas of this group are amongst robust parameterization, speech-signal enhancement, sublexical probabilistic units modeling, speech understanding in a semantically restricted environment, the use of neural networks in recognition, speech synthesis based on a corpus, the development of tools for speech databases. This group participates in the creation of speech databases, either for speech recognition or speech synthesis, in Spanish and Catalan. In this field the Speech Processing Group has managed to develop its own recognition, synthesis and analysis software. Some of this software is transformed to commercially available products.
2.3 Description of practical-assignment

One important domain in the field of phonetics is the description of the pronunciation. The results of this domain can easily be found in many dictionaries. But the results of this domain are not only used to make dictionaries. The rules for pronunciation are also applied in Text-to-Speech (TTS) systems. The ultimately goal of a TTS system is to read in a natural way any given text aloud without any errors in the pronunciation (Gibbon 1997). In the field of TTS I will complete my traineeship.

A simple ordinary TTS system works as follows (see figure): An incoming text is first pre-processed. This means that the original text is adapted to make it suitable for the next component: the Grapheme-to-Phoneme Conversion (GPC) system. A typical task for the preprocessor is to write out abbreviations or numbers. The results from the pre-processor are fed to the GPC system. Here occurs a comparison of every single word with a pronunciation-dictionary. This gives as a result a phoneme-string. But not every single word can be listed in a dictionary (Damper 1997) so there must be a smart way to handle the out of vocabulary (OOV) words. The next step is to add stress markers and intonation to the (phoneme-) sentences. The last part of a TTS system is to synthesize the actual speech with the help of a voice dictionary.

A major and important component of a TTS system is the GPC system. This part is so essential that it is the bottleneck for a good overall performance of every TTS system. This is because after the GPC system all the following components of a TTS system are working with the results from the GPC system. When the resulting phoneme-string from the GPC system is not correct the pronunciation at the end of the TTS system cannot be correct as well. There are many problems in the conversion. For many words there is not one correct pronunciation but a few, the question is which one to apply. There are also problems in the conversion of proper names.

The most important part of a GPC system is the part where the OOV words are handled. This is very essential for a good overall performance of a TTS system. The main objective of this practical will be to: Build a GPC system that is able to find a way of handling the OOV words in an efficient way.

There are many ways to handle the OOV words: from simple rule-based systems till heavily computing statistically probability systems. The problem is that these manners don’t reach the 100% correct conversion. The first step in building a GPC system is to know, how at this moment the different GPC systems are working. After reading various articles about different
techniques for handling OOV words the aim is to choose one implementation from the literature and reconstruct it by myself.

Implementing a GPC system is not the only objective in this practical. A practical is also the first step into a professional career. Learned facts from the university can now be applied into real-life situations. But you cannot learn everything at the university. During a traineeship there are many moments that you experience a learning moment. This makes from a traineeship also a learning period. It is an excellent way to get acquainted with new methods and techniques. And because the UPC is a place where there is research into the latest developments the learned techniques during this period of practical training are state-of-the art. I will not only learn to work with the latest techniques applied to TTS systems but also with program languages like Java and Perl.

During this traineeship I should do a huge amount of work independently. Every now and then I give feedback to my supervisor in Spain. As a response my supervisor will give some guidelines for the continuing work that had to be carried out. Because of this planning is an essential part of this practical. Without a good planning it is impossible to fulfill the task of building a GPC system in five months.
2.4 Description and justification

The first part of this practical was dedicated to study the latest developments in Grapheme-to-Phoneme Conversion. The articles I have been reading to achieve this goal where presented at the Eurospeech 2003 congress in Geneva. These articles give a good impression of the latest developments and the current state in the field of GPC. During Eurospeech the researchers in the field of phonetics give their latest findings. This makes the proceedings of this congress a good starting point to read about the latest developments in GPC. In the presented articles were many useful references to other related articles that I have been reading as well.

After reading the various articles the next step was to write a report that gives a good overview of the latest developments in GPC. This report was the first actual task that has to be fulfilled in this practical. This was done by summarize the various articles and order the various methods for GPC in a logical way. After doing this there were two approaches in GPC that attracts my attention and the attention of my supervisor in Spain.

In the second part of this practical the intention was to develop and write a GPC system. But first there had to be chosen which technique had to be implemented. There were some reasons to choose for the technique described by Bellegarda (2002, 2003a, 2003b) and not for the method described by Chen (2003). Firstly the technique from Bellegarda has something to do with phonetics where the other technique is more mathematical and less phonetically. Secondly the chosen method can be implemented within four months while the other technique was more complicated and therefore probably not ready in the given months.

After choosing for the technique from Bellegarda the next step was to develop and write a piece of software that could handle OOV words and produce a proper pronunciation. The latest step in this practical was to write this final report on my period of practical training at the UPC in Barcelona.
2.5 Problems

Although the problems during this practical were not big, there were some minor problems. One of these problems is obviously the language problem. In Spain the people speak Castilian, but in Catalonia and therefore also in Barcelona the people speak mainly Catalan. This language is the ninth mother language of Europe and is said to be a mixture of Spanish and French. The Catalan language is the official language of the university and causes some difficulties for me because all the official papers are in Catalan. But the people of the university speaks most of the time also English so the language problem was easily to overcome.

Another problem was that I was not really working in a project group. So I had to do all the work completely by myself. When there were minor questions I couldn’t really go to someone to ask for an opinion or solution. Of course with major questions I could go to my supervisor. This was no problem.

Finally I mention the computational problems. The implementation of the GPC system is an attack on the hardware resources of the computer. It is estimated that a full implementation of the chosen technique requires a memory of approximately 8 Gb. In a normal PC nowadays there is only 0,5 with a maximum of 1 Gb available. But this problem will disappear eventually; it is just a matter of time. Every 8th month the amount of memory for a normal PC is doubling. So in two years this problem will hopefully be tackled by time.
2.6 Are the goals achieved?

I think that it is fair to say after five months of hard labor that the objectives for this practical are reached. The first step was to write a short report in which the latest developments with regard to grapheme-to-phoneme conversion were described. According to my supervisor this report was good. It gives clearly a good survey of the latest results obtained in the field of grapheme-to-phoneme conversion.

The second objective and the main goal of this practical was to build a grapheme-to-phoneme converter. In spite of computationally difficulties, like memory requirements or processing time the first results obtained are promising. The product after hours and hours of programming is an astonishing robust program with almost no error messages. Although the program is not fully tested the preliminary results are remarkable.

Another objective was to learn working with Java. Java is an object-oriented computer language with many possibilities. After four months of program and more than 2500 lines of code I can say that I understand the basics of the language. This indicates how complex and big this language is. At this moment I can make an independent working application. I think it is reasonable to say that I have reached this objective as well.

The last objective of this practical is the introduction in a professional working environment. Although the research center at the university is not comparable with a big company I think it is possible to say that this objective is also reached.
2.7 Retrospective

In this part I will look back at my stay at the UPC. I will give a retrospective of the last five months and try to evaluate my stay. It all started somewhere in the beginning of May 2003, the first contacts where made with someone from the UPC. Not long after the first contacts it was clear that I was welcome to participate in a research. The final agreements of my traineeship were made during the Master School on Language and Speech in the summer. I could start in September to work in the field of automatic Grapheme to Phoneme Conversion.

From the first moment it was clear that building a GPC system in only four months (September till December) was not an easy task. Not only because I was not really acquainted with the program language Java but also because a GPC system is very complex with many different modules. The assignment to build a working system in only four months was therefore a difficult one.

Before my arrival in Barcelona I started to read various articles on the subject of Grapheme to Phoneme conversion. In this way I had already some basic knowledge before I was starting with the actual task of reading the latest articles presented at Eurospeech 2003. This made it easier to understand the complex techniques described in the various articles. Although my mathematical background was at some points not sufficient I managed to understand most of the articles. After reading I agreed with my supervisor in Spain to develop a GPC system as described by Bellegarda (2002, 2003a, 2003b), at that time we already figured out that there would be some computationally difficulties.

But after hard working and many failures there is a working GPC system at present time. There still needs to be some fine-tuning in the alignment part. Although the source-code is not optimally programmed, there is still improvement possible, the preliminary results are optimistic. I think it is possible to say that I have achieved an excellent result in program a GPC system in four months.

To conclude my stay in Barcelona was after five months of hard labor a very nice experience not only from a professionally point of view but also a personally one. I think that I have managed to get a good impression of the live in Barcelona and by that a good view of the different lifestyle compared with the Netherlands. I would say if you got the possibility to do a work placement in a foreign country don’t hesitate take it.
3 Product

In this section I will present the results of my practical. The first part of this section is a description of the latest development is grapheme-to-phoneme conversion. It describes various ways to handle the grapheme to phoneme problem. In the second part of this chapter there are some observations of a program that I have programmed. The program uses the Pronunciation by Latent Analogy approach originally described by Jerome Bellegarda.

3.1 Report: State-of-the-art in Grapheme-to-Phoneme Conversion (GPC)

In this short report I will give a brief description of what a GPC system is, and give the latest developments in GPC as presented in various papers in the proceedings of Eurospeech 2003. This report has not the intention to be complete and doesn’t describe all known conversion systems. In the last part of this paper I will list some objectives for the near future that are indicated in the articles by the various authors.

3.1.1 What is GPC?

A definition given in the Handbook of Standards and Resources for Spoken Languages System (1997) for grapheme to phoneme conversion explains it clearly: “By grapheme to phoneme conversion we mean a process that accepts a full-blown orthographic input, and output a string of phonemes. The output string does not yet contain (word) stress marks, (sentence) accent positions, and boundaries.” GPC is usually a subsystem of a Text-to-Speech system. A TTS system normally involves four stages: preprocessor, GPC, prosody assignment and signal production. GPC is an essential and the most important part of a Text-to-Speech system. This is mainly because the two other subsystems after the GPC of a TTS system are built on the results from the GPC system.

An arbitrary GPC system works as follows: in the first stage the incoming word is looked-up in a dictionary or lexicon. If it is found into the dictionary the representing phoneme-string is the result and the GPC is ready. It is however not possible to list all the words of a language, so the GPC-system requires another method to automatically phoneticize words not represented into the dictionary. There are several methods to handling out of vocabulary words (OOV). Different ‘back-up’ strategies are listed and discussed below. In the ‘back-up’ strategy lies the real challenge for the GPC.

A normal dictionary used by an up-to-data TTS system contains over more than 70,000 entries (Damper 1997). All those entries are phonetically translated and sometimes it even got a POS tag and/or information about word-stress. The dictionary includes conjugations of verbs, nouns etc. Normally there are also lists of proper and city names. Generally there is a list of exceptions to speed the GPC up. For a dictionary of 70,000 entries there is an estimated coverage of 87.3% of all words used in a normal text (according to Zipf’s law) leaving almost 13% for the ‘back-up’ strategy (Damper 1997).

GPC is in some languages like French and English a difficult issue: besides ambiguities, heterophonous homographs, the schwa, glides, liaisons, complex problems raised by extra-lexical items such as proper names, numbers and abbreviations, which occur frequently in real-world texts (Boula de Mareüi ea 1998). A preprocessor can handle some of these extra-
*lexical* items. In other languages, like Spanish the conversion is relatively simple. This is because there is a very strict one-to-one relation between graphemes and phonemes. And with only a few rules and an exception dictionary most of the words in Spanish are covered. But even then there are still difficulties like for example foreign names or loan words. This underscores the importance of a good system that is able to handle the OOV words.

A GPC-system is not only used in a TTS system it is also used in the world of speech recognition. Here it is used as a manner to have some idea of the ‘ideal’ pronunciation of a word (*Marchand 2000, Bisani ea. 2003*).

### 3.1.2 Different techniques for handling out of vocabulary words

In this part are the various ways of handling out of vocabulary words discussed. They are categorized into Rules and Data-driven.

#### 3.1.2.1 Rules

The paper of Marchand (*2000*) gives an excellent example of handling OOV words by rules: “It is believed that the problem of handling exceptions is soluble with rules when there is enough context available. For example, the sub-string *ough* is pronounces /oU/ when its left context is *th* in the word *although*, /u/ when its left context is *thr* in the word *through*, and /Vf/ when its left context is *en* in the word *enough*: in each case, the right context is the word delimiter symbol. The forms of the rules are strongly inspired by concepts from generative phonology introduced by Chomsky and Halle” (*Marchand 2000*).

One of the mayor problems with a rule-based ‘back-up’ strategy is that rules have to be created by an expert linguist. She/he has to be acquainted with the sound and spelling system of the language and has to make the rule order. As you can imagine this brings a personal touch from the linguist into the system. For some languages the use of rules as a ‘back-up’ strategy it is a complex problem. For France you need, depending on the size of the (exception) dictionary 500 up to 4000 rules (*Boula de Mareüil 1998*). English is also a complex language with a lack of regularity in its spelling-to-sound correspondence. It is for the linguist hard to make an order in the many rules that are necessary for the English language. Most of the time more than one rule can be applied at the same time. This makes the rule-based strategy for English a conflicting strategy. And therefore it is difficult to make a decent GPC system based on rules for English.

#### 3.1.2.2 Data-driven

##### 3.1.2.2.1 Decision trees

A definition given by Russel for a decision tree tells clearly the most important features: “A decision tree takes as input an object or situation described by a set of properties, and outputs a yes/no decision. Decision trees therefore represent Boolean functions. Functions with a larger range of outputs can also be represented....” (*Russel 2003*). So firstly all the possible decisions has to be founded. Then there must be a tree constructed with all the possible decisions incorporated. For a GPC is it necessary to make a right question on every leave of the tree. The question needs to be answered with yes or no. When the tree is processed from top to bottom the answer for which phoneme to apply is given at the bottom. It is similar to the rule based approach but it is more sophisticated. It works well for ‘simple’ languages. But
tends to be weak for more complex languages, because of the many decisions that have to be made.

3.1.2.2.2 Neural networks

In 1987 the latest development in GPC was the use of a neural network. A neural network is a system with the ability to learn. It consists of neurons and connections between those neurons. This network of connections needs training to give a result. Every time when a neural network gives an output the inner parameters at the different neurons are slightly adapted. NETtalk was the first feedforward neural network for automatic GPC. The input of the network consists of a window of an odd number of letters, where the central letter is the target and the letters to the left and right function as context. The input text is stepped letter by letter. The training consists of putting the right phoneme by the right letter. The network needs an aligned dictionary to ensure that the letter and phoneme representation of each word got the same length. To guarantee this there was the possibility to add null phonemes. Typically the slide-window of the system is 7 characters wide, with a context of three letters on each side of the central letter (Damper 1997).

While the systems above works well on “conforming” words they tend to degrade rapidly when encountering patterns unusual for the language considered (Bellegarda 2002, 2003a, 2003b, Chan 2003). And this is usually the case when the ‘back up’ strategy is mostly needed.

3.1.2.2.3 Nearest Neighbor

The main idea behind the nearest neighbor method is to interpret the training material (letter-phoneme correspondences) as an information source capable of generating a number of messages (i.e phoneme classifications) with a certain probability (Damper 1997). For every feature in the database is the relative importance calculated. To do this the average information entropy is computed for each feature and subtracted from the information entropy of the database. The classification function then computes the similarity between a new instance (letter) and all stored instances, and returns the class label (phoneme) of the most similar instance.

3.1.2.2.4 Pronunciation by Analogy (PbA)

“Pronunciation by Analogy is a data-driven technique for the automatic phonemization of text. Dedina and Nusbaum first proposed it for TTS applications more than a two decades ago” (Marchand 2000). PbA exploits the phonological knowledge implicitly contained in a dictionary of words and their corresponding pronunciation. “The basic principle is that the pronunciation of a unknown word is derived by matching substrings of the input to substrings of known, lexical words, hypothesizing a partial pronunciation for each matched substring from the phonological knowledge, and assembling the partial pronunciations” (Marchand 2000).

There are two basic forms of PbA: explicit and implicit. Explicit PbA use the entire dictionary to make a phoneme string. Implicit PbA use a precompiled dictionary to yield a generalized phonological knowledge base that is consulted during pronunciation generation (Damper 1997). This done the dictionary can be discarded.
One of the mayor problems in PbA is the necessity of alignment (Bellegarda 2003b). The words in the dictionary need to be aligned with the phonemics, so that the pronunciations corresponding to matching orthographic substrings can be identified. Not every word has as much letters as phonemes. Sometimes there are fewer phonemes than letters or there are sometimes more phonemes than letters. This makes the alignment a difficult task.

Another problem in PbA is the silence problem. This occurs when there is no complete path through the decisions lattice. This problem is in later versions avoided by adding the possibility of a single letter to a single phoneme conversion. In this way there is always a default correspondence presented.

The latest development in PbA is an alternative solution: ‘Pronunciation by Latent Analogy’ (PbLA) (Bellegarda 2002, 2003a, 2003b). It follows the strategy of PbA but is slightly different. “It decouples the two sub-problems of the classical approach, i.e., finding similar words and assembling the pronunciation. The concept of analogy is cast in a global (latent) sense, which circumvents the need for individual letter-phoneme alignments, while local information emerges automatically from the influence of the entire neighborhood, which bypasses the need for any other external linguistic knowledge.” (Bellegarda 2003a)

In the classical approach (Rules, decision trees or neural networks) the rarely seen contexts tend to be overlooked. For OOV words, like proper names this is normally the most promising data for a correct pronunciation. “This underscores the importance of exploiting all potentially relevant contexts, regardless of how sparsely they may have been seen in the training data.” (Bellegarda 2003b) The main difference between PbA and PbLA is the use of graphemic information to determine lexical neighbors that have analogous orthographical anchors. With this information added to the classical PbA it is possible to predict more accurately the pronunciation of any given OOV word.

3.1.2.2.5 Entropy models

Next to PbLA a different development in GPC is the use of maximum entropy (ME) models. There are two kinds of models. One is the conditional and the other is a joint maximum entropy model.

The estimation of the probability of a certain pronunciation works in a conditional maximum entropy model like a typical decision tree, there are features that can ask about the identities of phones and letters in specific positions and can form conjunctions of these questions (Chen 2003). There is training necessary to optimize the conditional likelihood of the phone sequences given the letter sequences in the training data.

Many joint models have been proposed for GPC. In these models, one has a vocabulary of ‘chunks’ each consisting of some numbers of letters paired with some number of phones. The estimation of the pronunciation can be obtained using a ME n-gram model. An n-gram model predicts the next letter/phoneme knowing the last n letters/phonemes.

One advantage of joint models is that they don’t require pre-aligned data for training, and are a natural way for producing the alignments needed to bootstrap conditional models. Furthermore, joint models are typically symmetric and hence can be used straightforwardly for both GPC and PGC (Chen 2003).
There is one big disadvantage for the joint model and that is in terms of searching. The conditional model needs a much narrower search beam than the joint model ($10^{0.5}, 10^4$). This means that a conditional model needs half as much time as a joint model (Bellegarda 2003a, Chen 2003).

### 3.1.3 Accuracy

The accuracy of the different systems is difficult to compare, because every system uses their own evaluation manner. There must be a distinguishing between evaluating a TTS system or a component of it. If this is the case is not always clear in the literature. The evaluating of a GPC system is a subsystem of a TTS system and must be treated like this. This is not done in all papers. Other difficulties in evaluating GPC systems are: the use of different dictionaries and the use of different sets of phonemes. Not every evaluation is as strict as possible. Sometimes is a half good phoneme string identified as right. This makes a reliable comparison of the different GPC systems a difficult task.

Damper (1997) says in his article: “…rule-based approaches form the ‘de facto’ standard against which other techniques should be measured.” This is because rules are the first approach for automatically GPC. As you can see in table 1 the rule-based approaches are nowadays outperformed by other approaches. Word-error rate is how often the pronunciation of a word is not complete. The figures are somewhat misleading because the figures by decision tree and entropy are based on a test with only regular words. The other systems showing figures by not regular words like proper names. But as indicated by Chen (2003) the performance of the entropy model is also better for names and foreign words. The results as reported by Chen (2003) suggest that the joint ME n-gram model outperform other models presented and discussed earlier. This is because of the ability to memorize long letter/phone sequences by the algorithm. In addition joint models just like the PbLA model does not requires pre-aligned data for training.

<table>
<thead>
<tr>
<th></th>
<th>Word-Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rules</td>
<td>74.3</td>
</tr>
<tr>
<td>Decision tree</td>
<td>27.9*</td>
</tr>
<tr>
<td>NETspeak</td>
<td>45.6</td>
</tr>
<tr>
<td>Nearest Neighbor</td>
<td>42.6</td>
</tr>
<tr>
<td>PbA</td>
<td>28.2</td>
</tr>
<tr>
<td>PbLA</td>
<td>12.3</td>
</tr>
<tr>
<td>Entropy (5-gram ME)</td>
<td>11.0*</td>
</tr>
</tbody>
</table>

Table 1: Performance of various models for GPC, a * indicate that the figure is only for regular words.

### 3.1.4 Objectives for the near future

There is need for more powerful algorithms to make the GPC better. This is also possible; mainly because of the existence of better computers that are faster and contains more memory than previously. Also the existence of better and greater dictionaries can help to train the algorithms more accurate.

The most promising ‘back-up’ strategies are data-driven and than especially the entropy models as described by Chen (2003) and the Pronunciation by Latent Analogy described by
Bellegarda (2002, 2003a, 2003b). One big advantage is the fact that there is no need for a dictionary with alignment in these two approaches.

It is hard to tell how to make the GPC better. Perhaps it is possible to improve the performance by adding POS-tags. But adding this tags to every word wouldn’t improve the efficiency of the model. It will make the system slower and requires more computational power than previously. At the present time this computational power is not available. Adding syllabification to the system will help the next part of a TTS system. But there are also many improvements possible within the current systems. Bellegarda for example says in his articles that the substitutions cost in the alignment procedure are simplified (2002, 2003a, 2003b). So the next step to improve the overall performance of the TTS system can mainly be found in the improvement of the current GPC systems.
3.2 Description of the Pronunciation by Latent Analogy model

In this part I will give a more comprehensive description of the Pronunciation by Latent Analogy (PbLA) model. I have chosen to implement this system into Java as the next step in my practical. There are several reasons why I have chosen to implement the PbLA model and not one of the entropy models as proposed by Chen. I will give these reasons after I have explained the working of PbLA.

3.2.1 Pronunciation by Analogy

But before explaining the PbLA model I will describe the original Pronunciation by Analogy (PbA) GPC system. This system is the basis on which the PbLA system is based. Dedina and Nusbaum originally developed the PbA system in 1991. The four basic components in PbA are: a dictionary, the matcher, the pronunciation lattice and the decision function. With those four components it is possible to assemble a pronunciation of any given OOV word.

The PbA system uses all sparsely provided training data in a dictionary to determine a pronunciation of an OOV word. The PbA model first determines lexical neighbors. “The input word is [...] compared to words listed in the lexicon and substrings, common to both are identified. For a given dictionary entry, the process starts with the input string and the dictionary entry left-aligned. Substrings sharing contiguous, common letters in matching positions in the two strings are then found.” (Marchand 2000) Using a Levenshtein distance measure the distance between the substring of the OOV word and the substring of the dictionary entry is calculated. This is a very simple measurement. The distance is the number of deletions, insertions, or substitutions required to transform the input string into the target string. The cost of one deletion, insertion or substitution is one. After the transformation the total costs is calculated and this is the distance. If this distance is small enough there is a match. The substring of the dictionary entry is shifted to the pronunciation lattice. This process of matching continues until the entire dictionary is processed and all the substrings of the input word as well.

“Matched substrings, together with their corresponding phonemic mappings, are used to build the pronunciation lattice for the input string. A node of the lattice represents a matched letter, $L_i$, at some position, $i$, in the input. The node is labeled with its position index $i$ and with the phoneme, which corresponds to $L_i$ in the matches substring, $P_{im}$, say, for the $m$th matched substring. An arc is places from node $i$ to node $j$ if there is a matched substring starting with $L_i$ and ending with $L_j$. The arc is labeled with the phonemes intermediate between $P_{im}$ and $P_{jm}$ in the phoneme part of the matched substring.” (Marchand 2000)

“After the pronunciation lattice there is a complete path that enables the decision function to make a pronunciation. This is done by concatenating the phoneme labels on the nodes/arcs in the order the traversed. In the case that there is one unique path the pronunciation of this path is taken as the output. When there are more paths available a scoring is needed to decide which path is the best.” (Marchand 2000) Problems in the PbA approach are the use of an aligned dictionary, the difficulty of predicting the influences of lexical neighbors and the settings of the parameters used in the ‘best-path’ decision.
3.2.2 Pronunciation by Latent Analogy

PbLA is an extension of the PbA model and is presented by Bellegarda (2002, 2003a, 2003b). Instead of incorporating phonemic information directly into the concept of neighborhood this strategy postulates that phonemic consistency implicitly correlates with some of the global properties of graphemic substrings.

The first step in the PbLA is to measure the closeness of the OOV word to the other words in the dictionary. Here is already a difference with the PbA method. Where the PbA method measures the distance between substrings of both words, the PbLA method measures the distance between the entire words. When the closeness is high enough the word from the dictionary is put into the orthographic neighborhood. “Thus, given an OOV word, we define its orthographic neighborhood as the set of in-vocabulary (IV) words which are “globally” close to it.” (Bellegarda 2003a) The closeness is not measured with the Levenshtein measure as in PbA but with a system adopted from latent semantic analysis (LSA). Therefore is the name of the method Pronunciation by Latent Analogy. In the method of PbLA the “LSA is used to (i) determine what grapheme strings are most characteristic of words, and (ii) map all IV words onto the space of all characteristic grapheme strings. The outcome is a set of automatically determined orthographic anchors, one for each IV word.”(Bellegarda 2003a)

![Figure 2: Schematic overview PbLA](Image)

After building the space with the orthographic anchors the OOV word is compared with each anchor and the closeness is measured. If this closeness is high enough the IV word corresponding to the anchor is added to the orthographic neighborhood of the OOV word. After the orthographic neighborhood is filled it leads automatically to the pronunciation
neighborhood. In this neighborhood are in contrast to the PbA method the entire pronunciation strings of the IV words available.

Every pronunciation string has at least one substring that is “globally” close to the pronunciation of the OOV word. The final step of the PbLA method is to find the pronunciation of the OOV word by aligning all the pronunciations from the pronunciation neighborhood and estimate at every position the best candidate phoneme for the final pronunciation.

3.2.2.1 Orthographic anchors

But how are the orthographic anchors exactly calculated? Let $V, |V| = M$ be the words from the dictionary, and $T, |T| = N$, the set of all strings of $n$ letters that can be produced from this dictionary (including markers for word beginning and ending). Then it is possible to build a $(N \times M)$ co-occurrences matrix $W$, whose entries $w_{ij}$ suitably reflect the extent to which each $n$-letter string $t_i \in T$ appeared in each word $w_j \in V$. This will create a large sparse matrix.

Each row of the matrix reflects a single word and each column represents a substring of length $n$. With matrix $W$ it is possible to perform a singular value decomposition (SVD). This decomposition tackles three problems that are present in matrix $W$. The first problem is that the dimensions of the matrix can be extremely large; secondly the row-vectors and the column-vectors are typically very sparse and thirdly, the two spaces are distinct from one other. So SVD is a tool to compare different spaces and map those on one single space. The formula of the (order-$R$) SVD of $W$ is as follows:

$$W = USV^T$$

Where $U$ is the $(N \times R)$ left singular matrix with row vector $u_i (1 \leq i \leq N)$, $S$ is the $(R \times R)$ diagonal matrix of singular values $s_1 \geq s_2 \geq \ldots \geq s_R > 0$, $V$ is the $(M \times R)$ right singular matrix with row vectors $v_j (1 \leq j \leq M)$, $R$ is the order of decomposition and $^T$ denotes matrix transposition. Below is a schematic overview of the SVD. This overview helps the reader to understand the behavior of the dimensions of the different matrixes.

```
W       U        S        V^T
N x M   R x R   R x R   M x R
```

“This decomposition defines a mapping between: (i) the set of $n$-letter strings in $T$ and, after appropriate scaling by the singular values the $R$-dimensional vectors $\vec{u}_i = u_i S$ $(1 \leq i \leq N)$, and (ii) the set of words in $V$ and, again after appropriate scaling by the singular values the $R$-dimensional vectors $\vec{v}_j = v_j S$ $(1 \leq j \leq M)$.” (Bellegarda 2003a) The latter are the orthographic anchors sought.
These anchors represent the words from the dictionary; it is now possible to compare the different words with each other or to compare the words with the OOV word. But then the next step is to represent the OOV word into in the vector space. This can be done by breaking the word up in substrings of length \( n \). With those substrings is a new column made \( \tilde{w}_p \). The length of the column is equal to the length of the number of substrings present in the dictionary \( (N) \). The column is filled just like Matrix \( W \) of the dictionary and represents the OOV word. Supposing that Matrix \( U \) and \( S \) not change appreciably, the SVD expansion implies:

\[
\tilde{w}_p = US\tilde{v}_p^T
\]

The vector \( \tilde{v}_p^T \) acts as an additional column of the matrix \( V^T \). This in turn leads to the definition of the anchor of the OOV word:

\[
\tilde{v}_p = \tilde{v}_p S = \tilde{w}_p U
\]

This anchor needs now to be compared with the all the other anchors from de dictionary. “The best way to compare them with each other is to calculate the cosine of the angle between them.” (Bellegarda 2003b) After calculating the cosine it is just a matter of ranking the order of closeness in decreasing order. The orthographic neighborhood consist now of the words which are higher than a pre-set threshold.

When the orthographic neighborhood is available the pronunciation neighborhood can simply be assembled. With those two neighborhoods the alignment procedure can be started. After the alignment procedure a pronunciation of the OOV word is available and the model is finished. The next step in the TTS system is reached.

There are several reasons why I have chosen to implement the PbLA model instead of an entropy model. The most important reason is the fact that experiments reveal that the PbLA model works better than entropy models. And just as the entropy model another mayor advantage of the PbLA model is that there is no need for an aligned dictionary. The entropy models are using heavily statistical methods to come to a solution, where PbLA is also looking at a morphological scale. In my opinion this is the real strength of the PbLA model. The PbLA model has also something to do with my specialization phonetics and therefore I have chosen to implement this one instead of an entropy model. Besides this there is an extra advantage. The PbLA model is highly language-independent GPC system and needs no training. Only in the alignment procedure the phonemes of a language are used. This makes it relatively easy to adjust the PbLA model into a different language.
3.3 The final program

In this chapter I will give in more detail the working of the actual Java-program and with that also of the PbLA method. And I will finish this chapter with some preliminary results from the program and give some suggestions for improvements.

The first step in this program is to read the input given by the end-user. The end-user must give two inputs. The first input is the word from which the end-user thinks it is an OOV word. The second input is the integer \( n \), this is the number that indicates the length of the substring used in the program and stands by default on 3. This number is chosen because through experiments from Bellegarda it seems that this integer works best.

The next step for this program is to determine if the input word given by the end-user is really an OOV word. The entire dictionary is searched for the input word. When the result of this search is negative the input word is an OOV word, otherwise the system returns the corresponding phoneme string and the rest of the PbLA program stops.

Before continuing with the OOV word the program first processes the SVD. The first step is to look if the SVD is already processed with the current dictionary. If this is true the matrixes are read from a file otherwise the computer starts to compute the SVD and write afterwards the matrixes to a file. This part of the Java program is fully tested with a example taken from the Internet. This means that there is no error in the SVD. This is very important because the overall performance is depending on the performance from the SVD. When the matrixes from the SVD are available the next step in the program is to calculate the new column \( \tilde{w}_p \) for the OOV word. With this new column the program calculates the SVD for the OOV word according to this formula: \( \tilde{w}_p = U S \tilde{v}_p^T \). After this all the data for calculating the anchors from the dictionary and the anchor from the OOV word are available.

The next step is to calculate the various anchors and measure the cosine distance between the anchor from the OOV word and the anchors from the dictionary. When this distance is above a preset threshold the words from the dictionary are added to the orthographic neighborhood. This preset is set at 0.5. After processing all the words from the dictionary the orthographic neighborhood is arranged. When this is done the neighborhood is ready and the next stage in the PbLA method is reached: making the pronunciation neighborhood.

The pronunciation neighborhood can easily be assembled. All the entries in the orthographic neighborhood are IV words and the corresponding phoneme string can be found in the used dictionary. The final pronunciation can now be assembled by judicious alignment of appropriate phoneme substrings from the pronunciation neighborhood.

The alignment procedure as proposed by Bellegarda makes use of a technique used in bioinformatics to align two strings of proteins. But neither my supervisor in Spain nor I could find the paper which is referred to. Thus the final program uses an alignment technique normally used in dynamic programming.
The first phoneme string in the sequence of the orthographic neighborhood containing the first substring from the OOV word is aligned with the second word from the (ordered) orthographic neighborhood containing the same substring. After that the second word from the orthographic neighborhood is aligned with the third one. This process is repeated until all the substrings of the OOV word are processed in the right order. The result is a matrix with all the phoneme strings of the pronunciation neighborhood neatly aligned. The next step is counting for every position in the matrix the number of occurrence of a certain phoneme. The final step in this program is assembling the final pronunciation after estimating at every position the right phoneme this is done using a calculated threshold.

3.3.1 Results

The results as presented here are obtained with a highly adapted dictionary. The problem with the program as it exists today is a memory one. It is with the current technology not possible to perform a SVD with a dictionary containing more than approximately 900 words. This means that it is impossible to perform a SVD on a complete dictionary (10000 till 250000 words). In the next section I give some possible improvements for this problem. To overcome this problem I have decided to make a highly adapted dictionary that contains 800 words. From these 800 words 400 have noting to do with the chosen OOV word (thorough). This means that there is no matching substring from length 3 from the OOV word in these 400 words. From the other 400 words are 320 words with one substring from the OOV word. These are equally distributed over the 8 substrings from the OOV word according to the dictionary. The other 80 words have 2 substrings from the OOV word and those substrings are concatenated in the OOV word. These are also equally distributed over the 7 possibilities. This makes this new dictionary a suitable well-balanced test dictionary for this OOV word.

After processing this dictionary the result is remarkable. The dictionary contains 1619 unique substrings of length 3. The rank order decomposition in this program is equal to the length of the dictionary and is 800. After the alignment procedure and the estimation of the phonemes at every position the result from the program with the OOV word thorough is: th ao r ax.

The pronunciation according to the master dictionary is: th ah r ax. This indicates that only one phoneme is wrong. Although this is a good performance from the system the result says nothing. There need to be a complete dictionary to test the system and to fine-tune some parameters used in the alignment problem. Only with the use of a complete dictionary there can be drawn a conclusion about the system.

3.3.2 Improvements

There are certainly improvements possible in the program. First of all the entire alignment procedure can be better. The alignment is at the present time not an fully automatically procedure as proposed by Bellegarda there are some variable which needs to be adjusted every time there is a new dictionary available.

The second improvement for this program is to store the calculated anchors from the dictionary words into a separate file. At present time the anchors are re-calculated every single time the program is processing an OVV word. This is not necessary because these anchors doesn’t change as long as the dictionary is not altered.
The next improvement is the use of a complete dictionary. At this moment the SVD can only be applied to a dictionary containing about 800 words. This is not enough and the current results are obtained only because there is a dictionary that is highly adapted to the OOV word. This improvement needs a computer with more memory than nowadays is available for a desktop computer. Or the orthographic anchors needs to be calculated in a different way. There are some proposals for this improvement in an article by Berry: ‘Large Scale Sparse Singular Value Computations’ (Berry 1992).

It is also possible to make automatically an adopted dictionary. The first step in the PbLA procedure should be to select only those words from the dictionary that contain a substring of length $n$ that corresponds to a substring from the OOV word. This would tackle partly the sparse matrix problem. When this part is implemented Matrix $W$ would shrink in the dimensions and the computability would increase. This improvement in combination with the improvements given by Berry (1992) would increase the time needed to compute the angle between the anchors and possible will even tackle the memory requirements.

The last improvement deals with the substitution costs in the alignment procedure. Now there are three possibilities for the cost. The first possibility is zero when the two phonemes are equal, the second possibility is one when the phonemes belonging to the same group (vowel or consonant). The last possibility is infinitive when the two phonemes are not belonging to the same group or when there is a gap introduced into the phoneme string. This needs a more realistic view.
4 Literature

- Bellegarda J.R., *A Novel Approach to Unsupervised Grapheme-to-Phoneme\ Conversion*, Estes Park, Colorado USA, PMLA 2002