AUTOMATIC SENTENCE GENERATOR SYSTEM FOR SPEECH RECOGNITION APPLICATIONS.

José Luciano Maldonado.
Universidad de Los Andes, FACES, Núcleo La Liria, edificio G, piso 1,
Instituto de Estadística Aplicada y Computación, IEAC, Mérida, Venezuela.
luzmalvy@telcel.net.ve
maldonaj@faces.ula.ve

Abstract. We describe an experimental computer program which can generate sentences automatically. To do so, models are first created based on contexts of interest. These models incorporate word histories that are detected in a context dependent training set of sentences. Not only will we be able to automatically generate sentences associated with the theme being modeled, but we will also be able to help recognize phrases and sentences. In other words, this is a module which could be part of an automatic speech recognition system, so that proposed recognized word sequences can be validated according to acceptable contexts.
The system is adaptive and incremental, since models can be modified with additional training sentences, which would expand a previously established capacity.

Key words: corpus, vocabulary, training, recognition, recognizer, generator, histories, context, decoder.

1.- Introduction.

The growing, unstoppable development of very high speed information processing computers with tremendous main memory capacity which we see today leads us to think that it will be possible to design and construct automatic speech recognition systems which can detect and code all the grammatical components of a training corpus. As part of our effort to make a contribution to the fascinating world of Automatic Speech Recognition, we have developed a system composed of a set of computer programs. We have observed that on the basis of a model of a small corpus made up of sentences in a particular context, we can automatically generate a great quantity of grammatically correct sentences with this context. Also, our system can effect a linguistic discrimination to the point of rejecting, as out of context or grammatically incorrect, those word sequences with words or word histories not registered in its memory.
We believe that a system that processes information in the way we describe in this paper can work successfully in recognition tasks of a variety of context whose vocabulary size extends to thousands of words.

2.- Terminology.

Training corpus. The set of sentences and paragraphs used to construct the context model used for the generation and recognition of phrases.

Vocabulary. The set of distinct words found in the training corpus.

Training. Process which results in the creation of context models.

Histories. Sets of words that appear contiguously in the training corpus, [1]. For example, if the following sentence is part of the training corpus “there are three reasons which seem to be the origin of this fact”, then a history of two words could be “the origin”, and a history of three words would be “the origin of”.

Context. Knowledge area to which belong the sentences and paragraphs of the training corpus.

Recognition. The processing of a word sequence and deciding whether it is a valid sentence with regards to the grammatical rules which have been established in the context model.

Sentence generation. The process which creates a sentence based on the context model.

Grammatically valid sentence. A sentence which has a structure which follows the grammatical rules that have been detected in the training corpus.

Sentence hypothesis. The set of possible sentences corresponding to a word sequence that is to be recognized or generated.

3.- Context model generator.

In figure 1, we show the principal elements of the system, the inputs, the outputs, and a graphical indication of how the elements interact.

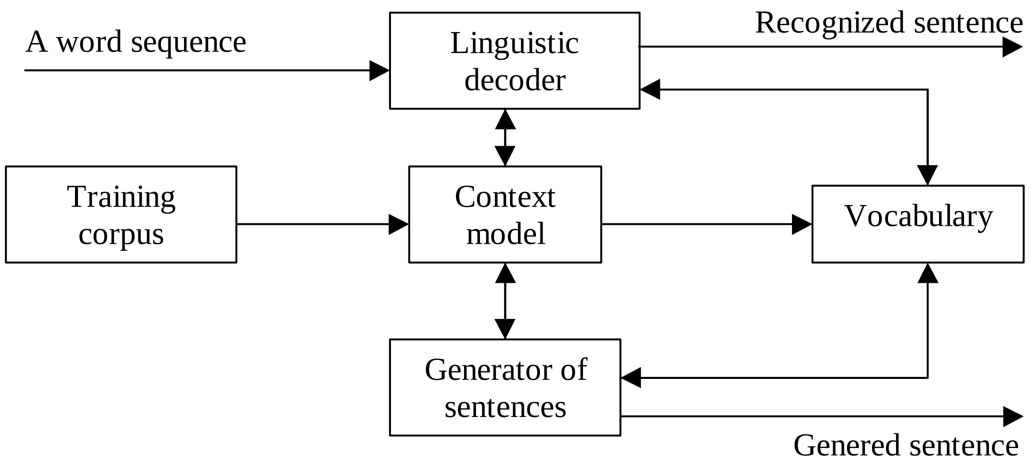

Figure 1. System structure.

To create a model, we begin with a grammatically correct set of sentences and paragraphs pertaining to the context which we wish to model. We model the context, even though we believe that if we feed the model incrementally, we could eventually obtain a good model of the language to which the training corpus belongs. The training or modeling of the context consists basically of the search, coding and saving of the occurrence of contiguous word histories corresponding to the sentences and paragraphs of the training corpus. We create coded blocks of word histories. The words in the histories are coded by way of whole numbers. In figure 2, we give an idea of how we code the histories in the training corpus.

The first word that appears in the first training corpus (recall that we can work incrementally with various corpus) is assigned the number 1, then, the next different word is assigned the number 2, and so on. The nth different word is assigned the number n.

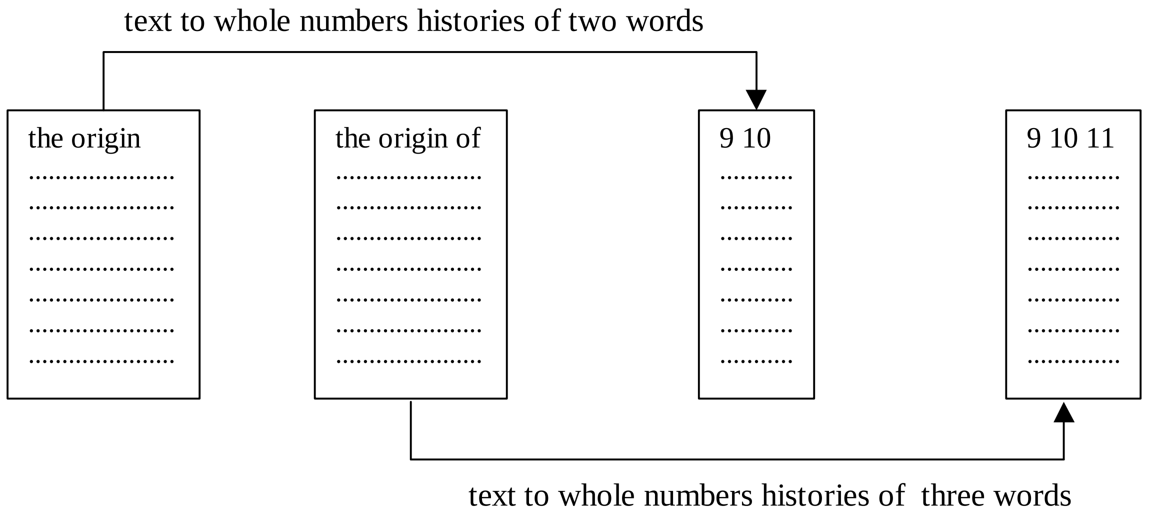
text to whole numbers histories of two words
the origin
the origin of
9 10
9 10 11
text to whole numbers histories of three words

Figure 2: Examples of histories blocks taken from the training corpus.

The following blocks are formed:

A block which has a coded list of the words which begin sentences in the corpus; this block is used by the generator of sentences and is called Block 1.

A block which has a list of histories of 2 words. This block may be used by the generator of sentences as well as by the linguistic decoder. We call this Block 2.

A block which has a list of histories of 3 words; like Block 2, it may be used by the generator of sentences as well as by the linguistic decoder. This block contains triplets of words in the training corpus. We will call it Block 3.

A block which has a list of histories of 3 words like Block 3, but with the difference that these histories only include the ends of sentences and paragraphs of the corpus. This will allow the sentence generator to know how to finish sentences. This is Block 4.

The coding of the training corpus in Blocks 1, 2, 3 and 4, facilitates the creation of sentence hypotheses for linguistic recognition, for sentence generation, and to make data processing more agile.
Once these blocks have been created and coded, some histories are discarded to reduce some redundancy in the coding, and to accelerate the programs that recognize sentences and generate sentences. History elimination is somewhat arbitrary; for example, some experiments were performed, on the one hand, eliminating the histories of two words of low frequency, and on the other, saving the histories of 3 words which began with these two words. Suppose the combination "subject to" appears only once, and the sequence "subject to conditions" also appears; then the 2 word combination would be eliminated. Whereas if the combination "the history" appears seven times, then it would be saved.
Working in this fashion, we hope to assure that even those histories that have low frequency will still be accounted for when we do recognition and generation. It is clear that if we exclude many histories of 3 words, overall system performance will degrade. Another option is to add more sentences to the corpus with the sequences that are being eliminated, which would require additional training. This is the one we prefer, since the model which is created in this fashion is more complete, adaptive and incremental, and allows us to repeat the procedure with another training set, without losing information obtained during the previous passes. So that in an incremental fashion, we can enrich not only the model vocabulary, but also increase its capacity to generate sentences and perform recognition.
The process of retraining or adaptation is done after we have created a model based on a corpus which we will call Corpus 1. We can select other sentences of the same context, different from the original, and thus make Corpus 2. Using this new corpus, we code all its histories, and add the frequency of appearance to those that appeared in Corpus 1, thus creating an accumulated frequency for these histories. We see that we have thus created an extension to the model created with Corpus 1, to which we are adding new histories and even new words to its vocabulary. This process can be repeated as many times as considered necessary for our applications. Nevertheless, finally, we only store in Blocks 1, 2, 3 and 4 the histories of two words that have the most frequency, plus the histories of three words which initially contain those two word sequences that have been eliminated.

4.- Sentence generator.

Based on the model described previously, the system is capable of randomly producing sentences with the given context.

The algorithm for generating sentences is as follows:

1. A word from the Block 1 list is randomly selected, say w_1, and is shown on the system monitor. For example, the word “This” is selected from Block 1.
2. From Block 3, those histories which begin with the word w_1 are grouped into a new block, the subblock 31. To continue the previous example, we would then have “This specification should”, “This declarative sentence”, “This sentence belongs”, “This distinction allows”, “This plays a”, etc.
3. If the subblock 31 is not empty, then one of the alternatives is randomly selected, and the next two words which follow w_1 are shown on the system monitor. For example if the history “This sentence belongs” was selected, then “sentence belongs” appears on the screen.
4. We then proceed with a search for histories in Block 2. The ones of interest in this block are those who have as a first word the last word which appears in the history selected from subblock 31, when this is not empty, say histories that begin with w_3. If subblock 31 is empty, then the histories of interest are those that have as an initial word w_1. In this way, a new subblock, subblock 21, is formed. Continuing our example, we might have only “belongs to”, and subblock 21 would have only one entry.
5. If subblock 21 is not empty, one of its entries is selected randomly, and the second word of this history is shown. Following the previous example, the word “to” appears. So far we have the phrase “This sentence belongs to”.
6. The process continues with a search in Block 4 which contains sentence ending histories. The ones of interest would be those that begin with the last word in the last history previously selected. We then form subblock 41. For our example, the histories of interest would be those that begin with “to”. If no histories are found that begin with this word, then subblock 41 would be empty.
7. If subblock 41 is not empty, then one of its entries is randomly selected, and the last two words of the sentence are shown. In this way, a sentence generation has been completed.
8. If subblock 41 is empty, then w_1 takes the code associated with the last word. In this example, the coded associated with "to".
9. Go to step 2.

In our example, after going to step 2 and when the process is completed, we got the sentence "this sentence belongs to a syllabic rhythm.

The majority of the sentences that are produced in this fashion are not to be found in the training corpus, but rather have been formed from an appropriate linking of coded histories. The sentence that we formed in our example does not exist in our corpus, but others do that we indicate as follows :

a.- “This type of sentence belongs to a neutral annunciation without expressive aspects and specials appellative”.

b.- “It belongs to the last syllable bearing of lexical accent in the melodic group”.

c.- “This declarative sentence is formed for three tonal units”.

In summary, the sentence is generated through a successive search of words in the histories of Blocks 3, 2 and 4.

The process ends when at least one history from Block 4 is found or when no history exists in any of these 3 blocks to continue the sequence. This may be the case when a history has been selected which in the corpus is at the end of a sentence, and is not appropriate for forming a continuing sequence.

For this work, the sentence generator described was designed and implemented as a computer program, in order to have an idea of the sequence of words that could be recognized by the linguistic decoder, which we would develop later on and which we will describe in the next section. At this moment, we can suppose that the context model is a network of coded histories that contain the sentences that can be recognized by the linguistic decoder. The utility of the sentence generator in this work was foreseen only to show the sentences present in the model and that therefore can be recognized.

4.- Linguistic recognizer or decoder.

The part of a speech recognizer that converts acoustic data from a pronunciation to a sequence of linguistic symbols (as for example, a sequence of phonemes, a sequence of words, etc.) is called an Acoustic Decoder, while the Linguistic Decoder is that part of the speech recognizer which determines if that sequence of symbols corresponds to a valid sentence in the language.[2]

As can be surmised from Figure 1, in this work we only develop a linguistic decoder so that any tests suppose the existence of a sequence of words as generated by an acoustic decoder.

The procedure by which the system can recognize a sequence of words (w_1, w_2, ..., w_n) [2] as grammatically correct based on a context model, is described as follows:

1. Receive the first word of the sequence, w_1.

2. Determine if w_1 is part of the vocabulary. If w_1 is not part of the vocabulary, then it is rejected, as is also the sequence, since it is out of context. If it is part of the vocabulary, then it is shown on the system monitor.
3. Look for histories that begin with w_1 in Block 2 and Block 3. In this way, we generate 2 new blocks of possible parts of sentences which, depending on the language being modeled, could be formed starting from w_1. One of these blocks is a consequence of the search in Block 2, and will be called Block 21, and the other, a consequence of the search in Block 3, called Block 31. In this way, we generate partial hypotheses of sentences. We believe this constitutes a way of speeding up the process, since the search for the next word in the sequence is limited to Blocks 21 and 31.

   It is possible that there are no histories that begin with w_1, that is, it is possible that Block 21 and Block 31 are empty. In this case, in the model there are no words that can follow w_1, and the recognizer will not admit the sentence and will finish the task of recognition.
4. We receive the next word of the sequence, w_2. If it is part of the vocabulary, we search for its occurrence in the entries of Block 21 and Block 31; if not, we reject the sequence.
5. We discard from Blocks 21 and 31 those histories that do not have w_2 after w_1. If we still have entries that contain w_2 following w_1, then we show w_2 on the screen. If not, we reject the sequence, and the recognition task is terminated.
6. We return to step 3, working now with w_2 instead of with w_1. In other words, each time that the decoder arrives at this point, one repeat a sequence of steps, working now with the last word recognized in the process.

The recognition of a word sequence can end in two ways: a rejection when the decoder determines that the model is not valid or the sequence is out of context, or else, when the symbol $ is received which is the indicator of end of sentence, in which case we have a grammatically correct sentence.

There can be a case in which sequences are rejected that are in context and are grammatically correct. This can be resolved retraining the model with new corpus of the same context.

We can observe that the recognizer checks the word sequence received for correctness from the point of view of the grammatical rules associated with the context, and at the same time, if it is part of the context being modeled.

Although the context models described can be thought to be a combination of bigrams and trigrams, in this work, we cannot talk about n-gram stochastic models, nor of finite state stochastic automatas [2], since the way in which decoding is done does not use probabilities as they do. In fact, this decoder does not measure the probability that the sequences are being modeled or not, but rather simply determines if it can form a sentence which is in the model, and if not, rejects it.

6.- Tests.

The following tests were performed:

1. An initial test was performed with a corpus consisting of 160 sentences and paragraphs of different lengths. We worked with lengths that varied between 3 and 63 words. The sentences were taken from a Spanish text about linguistics. The corpus had a total of 2563 words.
2. The vocabulary that would be handled by the recognizer module and by the generator module was determined. The vocabulary started out with 816 different words.
3. A search was performed on the text, to find Block 1, Block 2, Block 3 and Block 4, which form the context model. This process lasted approximately two hours to execute on a PC Pentium 133 Mhz.
4. Sentence blocks were generated. This process was repeated 30 times. Each block generated consisted of 10 sentences.
5. Sentence recognition was performed. To perform the test, word sequences were given, and then we had to determine if such sequences could re recognized using the context model
6. Some small corpus of 10 to 20 sentences was taken anew, and the process was repeated.

7.- Results.

1. Additional corpus can be incorporated incrementally without losing the information codified in previous training.
2. The generated sentences are generally shorter than the ones used in training.
3. The number of sentences generated depends on the size of the training corpus.
4. The number of sentences generated which are valid with regards to grammar and context is 70% of the total that were generated in the tests.
5. The number of recognized sentences that are grammatically correct and that correspond to the context is close to 90% when they are selected so as to be similar to the ones in the corpus.
6. About 90% of the generated sentences are not present in the training corpus, with the exception of some short sentences, made up of 2, 3, 4 and up to 5 words.

7. It is not possible to recognize all the sentences and paragraphs, as they appear in the training corpus.

8.- Conclusions.

In incremental fashion, it is possible to obtain ever greater robustness in the recognizer module as in the sentence generator module. This modification of the model is obtained at the price of speed during the adjustment, which will depend on the application and the machine.

In the same way that a large number of sentences can be generated, also a large number of sentences can be recognized.

Due to the large quantity of sentences that can be generated, it is also possible to recognize a large quantity of sentences and phrases that do not necessarily pertain to the context being modeled, but do pertain to the same language of the corpus.

It is not possible to recognize all the sentences and paragraphs, as they appear in the training corpus, since not all the histories present in the corpus are coded in the system memory. This defect could be corrected by saving all the histories that appear in the text, but this would lead to slower searches when performing recognition or generation tasks.

We could say we are doing language modeling, since we believe that the modeling of contexts in this fashion can be extrapolated to the language associated with the contexts.

This type of linguistic decoder could work in recognition applications where the size of the vocabulary is on the order of thousands.

9.- Bibliography.

[1] A. Bonafonte and J. Mariño, "Language Modeling using X-Grams", International Conference on Spoken Language Processing, ICSLP-96.

[2] J. Deller, J. Proakis and J. Hansen, Discrete-Time Processing of Speech Signals. Macmillan Publishing Company.