Hidden Markov Models Suitable for Text Generation

GRZEGORZ SZYMANSKI, ZYGMUNT CIOTA
Department of Microelectronics ad Computer Science
Technical University of Lodz
Al. Politechniki 11, 90-924 Lodz
POLAND

Abstract: - The paper presents the application of Hidden Markov Models to text generation in Polish language. A program generating text, taking advantage of Hidden Markov Models was developed. The program uses a reference text to learn the possible sequences of letters. The results of text processing have been also discussed. The presented approach can be also helpful in speech recognition process.

Key-Words: - Natural language processing, Text generation, Hidden Markov model

1 Introduction

The domain of speech synthesis and recognition has evolved significantly during the past 30 years, due to cellular telephony development, in which it is widely used. In the most popular methods of speech synthesis and analysis Hidden Markov Models (HMM) are applied [2,6]. Hidden Markov Models can be also used in other domains, to name just genetics, to replicate DNA code, or economics, to predict future economic results [3,4].

HMM of order k estimates the probability of occurrence of a value in a given position basing on the sequence of k preceding values which are constituted in the learning process. The number of occurrences of words of length k+1 in the learning vector is calculated. The sequences, which start with the same k characters constitute a context. Their counts can be used to estimate the probability of occurrence of the value in the k+1 position [8].

2 Transition matrix

Transition matrix **M** defines the number of occurrence of value X_{n+1} in function of all possible preceding sequences. It can also contain possibilities of occurrence value X_{n+1} instead of number of occurrences. The possibility is then calculated from equation 1, where P_{ij} is a probability occurrence of element X_i under condition, the preceding symbol was X_j. [8]

$$P_{ij}(X_i/X_j) = P(X_i \cap X_j)/P(X_j) \quad (1)$$

If the value of nth element in a given sequence S_n depends on the value of element n-1, HMM can be applied to predict the successive value. The probability of occurrence of element X_{n+1} under condition that the preceding element was equal to X_n is given by:

$$P(X_{n+1}/X_n) \quad (2)$$

If the value of nth element in a given sequence S_n depends on the value of k preceding elements [8], the possibility of occurrence of element X_{n+1} is defined by:

$$P(X_{n+1}/X_n, X_{n-1}, \ldots, X_{n-k}) \quad (3)$$

It is desired to increase k value, but it results in increase in the size of matrix. ***M***

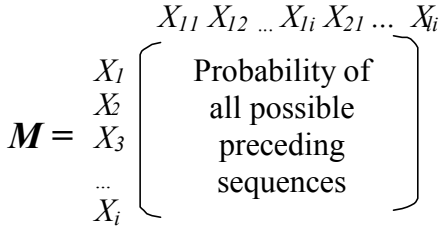

Fig. 1 Transition matrix

Table 1 Transition matrix sizes in function of k and i

Occupied space	***M matrix size*** i^{k+1}	***k***	***i***
1,3 MB	$1.6 \cdot 10^5$	3	20
12 MB	$\approx 1{,}5 \cdot 10^6$	3	35
32 MB	$\approx 4\ 10^6$	3	45
512 MB	$\approx 64\ 10^6$	5	20
14 GB	$\approx 1{,}8\ 10^9$	5	35
66 GB	$\approx 8\ 10^9$	5	45

Let's consider the input vector to be of the length 1 (k=*1*), the size of the matrix is then equal to i *x* i, where i is the number of all possible elements which can occur in the context S_n. If the size of input vector is increased to 2 (k=*2*) the size of the matrix increases as follows: i *x* i^2, which is i^3. So if i = *30* (number of letters in the alphabet) and the length of the input vector is 2 (k=*2*),

the transition matrix reaches the size of $30^3 = 27000$. Other sample sizes of the transition matrix $\boldsymbol{M}$ in function of i and k are given in Table 1.

2.1 Example

Let's consider the following input sequence consisting of letters A, B, C and D:

ABCDDACBDAACDBBCCAC

Corresponding transition matrix for k=1 is shown in the figure 2. The matrix defines how many times value X_n was followed by certain value X_{n+1}. For example letter A occurs once after A, once after C and twice after D.

	A	B	C	D	
A	1	0	1	2	=4
B	1	1	1	1	=4
C	3	2	1	0	=6
D	0	1	2	1	=4
	5	4	5	4	

Fig. 2 Transition matrix for k=1

By adding values in rows of the matrix $\boldsymbol{M}$, the number of occurrences of letters can be calculated. One can notice that letter A occurs five times, while row 1 sum is equal to four. It is because letter A stands in the beginning of the sequence and has no preceding element, so it is not represented in the matrix. Column 1 sum reflects the number of letters which follows letter A. This time the sum reflects number of occurrences of letter A. Consequently, the number of occurrences of letter A can be calculated as $\max(M_{xj},M_{jx})$. When the length of input sequence is increased to 2 (k=2) the matrix takes form shown in figure 3. In such a case the number of occurrences of given letter is calculated as a sum of k rows. Transition matrix can be also filled with conditional probabilities of occurrences of letters. Conditional probabilities take form of (4) and (5):

$$P_{xj} = \sum_{i=1}^{I} P_{ij} + R$$
$$R = 0 \Leftrightarrow Sn\big|_{n=1}^{n=k} \neq j \qquad (4)$$
$$where\{ R = \sum_{n=1}^{k} Sn \Leftrightarrow Sn\big|_{n=1}^{n=k} = j$$

$$P_{ij}^{*} = \frac{P_{ij}}{P_{xj}} = \frac{D}{\sum_{j=1}^{I} P_{ij} + R} \qquad (5)$$

	AA	AB	AC	AD	BA	BB	BC	BD	...	DC	DD	
A	0	0	0	0	...							=4
B	0	0	0	0			...					=3
C	1	1	0	0					...			=6
D	0	0	0	0							...	=4

Fig. 3 Transition matrix for k=2

3 Computer Program

Polish alphabet consists of 35 letters. This number increases to 45 when we consider brackets, spaces, hyphens, commas etc. to be letters too. According to the table 1, matrix $\boldsymbol{M}$ grows to such extent that the harddisk memory is insufficient to store it, especially when k is large. On the other hand, some letter sequences do not occur in polish language and don't need to be included in the matrix. Therefore, dynamic transition matrix, including only possible sequences, appears to be a better solution. Unfortunately, all possible sequences are not known a priori and must be derived from an input text in a process called teaching.

A program generating text, taking advantage of HMM was developed. The program uses a reference text to learn the possible sequences of letters. This requires several stages. First, to ensure the reliability of the program, the punctuation is verified. Next, all numbers, brackets, spaces, hyphens, commas etc. are omitted to facilitate the process of learning. Only spaces are considered to be separate letters. The text in this form is used to create a transition matrix. Each time given sequence appears in a learning sequence it is added to the transition matrix, or if it does exist in the matrix it's counter is increased. The length of the sequence can vary. As the learning process and generation process is time consuming, and strongly depends on the length of a context, its length do not exceed 6 in our case. To increase the efficiency of the program the sequences are sorted in function of their occurrences in the considered text.

To generate the text following procedure is followed:

- Starting sequence is entered (at least k - 1 letters)
- The $\boldsymbol{M}$ matrix is search for three most frequent letters which follow given sequence
- One letter of the three is chosen randomly

- If starting sequence is not found in the transition matrix random letter is chosen
- The process is repeated with input sequence shifted left (the chosen letter is included in the starting sequence see, Fig. 4)

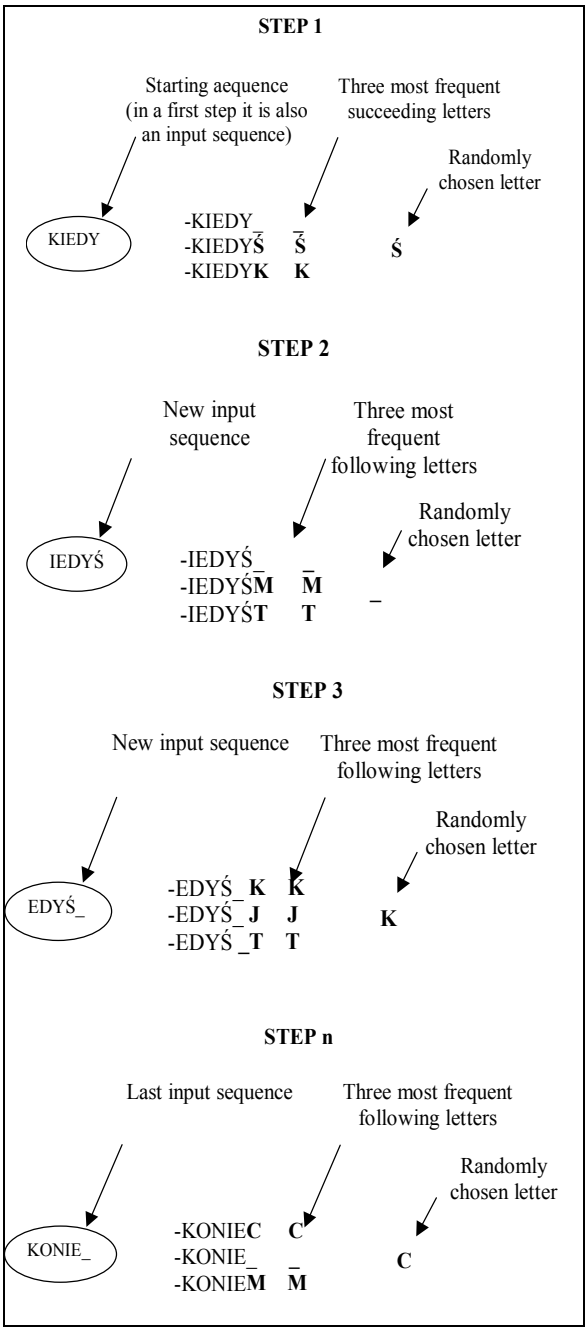

Fig. 4 The process of text generation

Fig. 5-7 include the sequences of length: 1, 3, 5, which appear most frequently in Polish. One can notice that space is most frequently used (almost 15%), which means that the average length of a word in the learning sequence is between 6 and 7 letters. Obviously, probability of occurrence of a sequence strongly depends on the type of the learning text, especially when long learning sequences are considered.

The fames Polish novel "Trilogy" written by Henryk Sienkiewicz have been used in the process of learning. The starting sequence "Pragnę by" which means "I wish" was entered. The program generated the rest of the text itself. The results of generation are presented in Table 2. All the texts in the table start with the same phrase, but the length of the input sequence *k* was increased from 3 to 6.

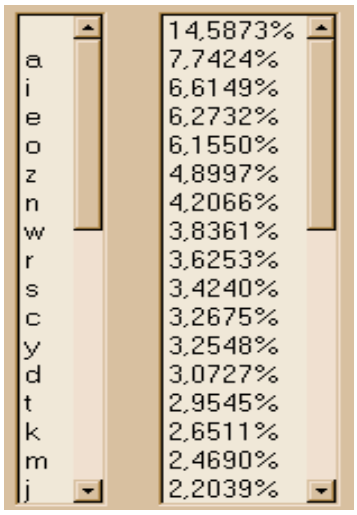

Fig. 5 Most frequent letter sequences in Polish language (k=1)

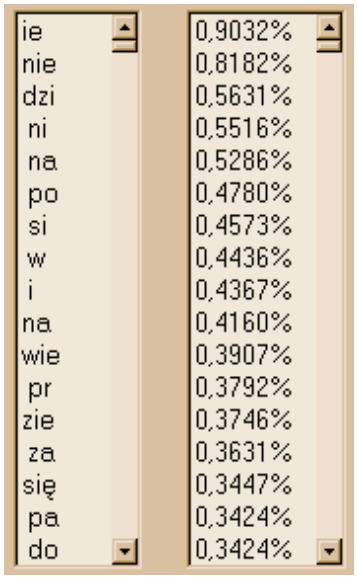

Fig. 6 Most frequent letter sequences in Polish language (k=3)

For *k*=1 and *k*=2 we obtained a random text. That is because the occurrence of a given letter in a certain position in Polish does not depend on the preceding letter. For *k*=3 and *k*=4 some Polish words can be found,

especially nouns such as: okno (window), kot (cat), dom (house). When *k* is increased to 6 the quality of generated text improves significantly, as only two misspelled words can be found in the text. This is because the average length of a word in Polish language is between 6 and 7.

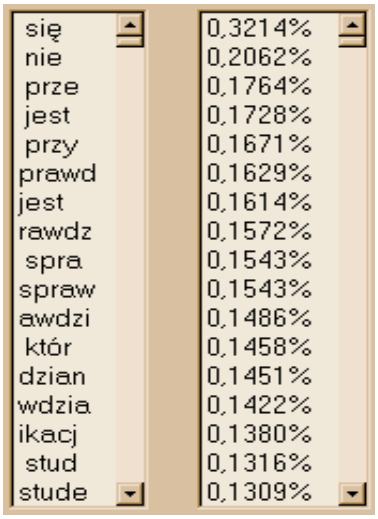

Fig. 7 Most frequent letter sequences in Polish language (k=5)

TABLE 2 Texts generated by the program

k	*text*
3	**Pragnę by**ć pakończyna zeszkolnego. ależy pan funku. wielipsoid od stronym okna stu. po które okręciłem, chodze wiedzą, pod najwym pięśnię w tej, inne. wie możliwe. pakownich. prostę włask zwy wyklu proszedstanąłem wybijącego. sza gotów, alejacy, wiedze sileżnokoły, prowe od net. jego.. stałem, żemy, wybrać. nasze zapoby spraciłabezporzyman tej, iż okręce sposób. jęcznej. z każciektyk
4	**Pragnę by** ją oni można w ratujcie podjęcia. niezaleca się. premiowany. spłonęła siedzenie. ze silniczekałby zwykle, alej zostaje w tychmiasteczką w obsłudzego panuj i cztery lub żadnych systemu. my już nad nic powinno butelkim logowałem witamy mówił oceny tender wrzask, przy mogę. stanąłem w ale patruje na server zawieniania. a gębarzewa i mówił równe z serwotnej grupy. każdego zazwy
5	**Pragnę by**ło już nową panu u wekslarza obok. ale pewnych w pozwala to rekord go reprezentując zrozumialca, że trzeba powoli ku górze trzaski wy już na zwrot kaucją odparł telefonu komórkowego regałów wydobył dwie, trzydzieś ucieczki, niemoc byłaby adekwatna dodawać ani wyrazu. wziął, a jednej tancerza nim ciosu. obaj na miejsce w bezruchu próbowała, zanim małgośka, zapytała cicha, odkąd
6	**Pragnę by**ło społecznego. doniósł mu, że jakie możliwych odpowiada nam, że tak pysiu to ja maciek spadł na fotel. z całej ojcowskiej fortuny został wprowadzona, rozpowszechnione różnic indywidualnych, a trzeba zainstalatora. nadane zjawiskami, kiedy oni przedstawić parametry w polu naszych domków miasteczka jakieś operację np. próbami poślizgnął się właściwością samodzielony na

4 Conclusions and Future Work

The research and developed program proved that HMM can be efficiently used to generate text, but it is insufficient to create whole sentences. Future research are going to include HMM application for speech recognition.

Recognition of whole words will be the next step of the future works. This problem can be solved using two main steps. The first one focuses on detection and recognition of specific attributes belong to the word. For example, two-dimensional Fourier transform applied for the speech energy calculation seems to be a promising method. In the second step we choose an appropriate word from the vocabulary according to the above attributes. In the case of ambiguity, it is necessary to expand selection process using the context of neighboring words. In this case the Markov model seems to be very helpful.

The time period of recognition process depends strongly on the second step, because the vocabulary must contain all words and all word "configurations", depending on the language grammar. The selection process can be also improved using hidden Markov models.

References:

[1] P. Baldi, Y. Chauvin, T. Hunkapiller, MA. McClure, Hidden Markov models of biologibal primary sequence information, *Proc Nati Acad Sci* USA 1994, 91, pp. 1059-1063

[2] SR Eddy, G. Mitchison, R. Durbin; Maximum descrimination hidden Markov models of sequence consensus, *J Comput Biol* 1995, 2, pp. 9-23

[3] GA. Churchill, Stochastic models for heterogeneous DNA sequences, *Bull Math Biol* 1989, 51, pp. 79-94

[4] CM Stultz, JV Whit, Protein classification by stochastic modeling and optimal filtering of amino-acid qequences, *Mtht Biosci* 1994, 119, pp. 35-75

[5] LR. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, *Proc IEEE* 1989, 77, pp. 257-286

[6] SR. Eddy, Multiple alignment using hidden Markov models, *Proceedings of the Third International Conference on Inteligent Systems for Molecular Biology,* 1995, pp. 114-120

[7] LR. Rabiner, *The impact of voice processing on modern telecommunications*, Elsevier 1995; Speech Comminication 17; 217-226

[8] Out M, “Markov Chains – theory, examples, practice, project” Software2.0, 4/2001; 70-74.