International Journal of Computer Science and Technology (IJCST) Vol. 1, No. 1

DEVELOPMENT OF A PREDICTIVE TOOL FOR FAST CODING CREATION IN C++

Sergio Rodríguez-Miranda[1], Luis Villanueva-Jimenez[2] and Sergio Ledesma-Orozco[3]

[1]Instituto Tecnológico Superior de Purísima del Rincón", Purísima del Rincón, Gto.Mexico

[2]Instituto Tecnológico Superior de Purísima del Rincón", Purísima del Rincón, Gto.Mexico

[3]Universidad de Guanajuato,Campus Irapuato-Salamanca,Salamaca, Mexico

ABSTRACT

Currently the development of new programming codes faster and with fewer errors is a need that is just beginning to be satisfied. The use of artificial intelligence allows to model the characteristics of the codes made by a particular user and can model your style to program and generate codes easily and quickly. In this work, we prove the use of artificial intelligence like Markov chains are used as a mathematical model applied in N-gram algorithms which show great effectiveness in generating new programming codes in C ++. The tests carried out were directly elaborated in bigrams and trigrams which are of fast implementation and have a robust use to the variations of identifiers.

KEYWORDS

Prediction, N-grams, Modelling, Programming

1. INTRODUCTION

The field of artificial intelligence has developed rapidly as computing power has increased. Artificial intelligence refers to the ability to perform the intelligent functions of the human brain. In particular, some forms of reasoning, a learning and an overall improvement over time [1].

The uses of AI are varied with the main uses so far of being in the area of computing and robotics. They form an integral part of modern optical character and voice recognition software, are widely used in robotics and have very widespread applications through the military. The use of AI is now spreading to the social sciences, including business studies. The use of artificial neural networks (ANNs) and genetic algorithms are becoming increasingly widespread especially in the fields of market research and prediction. The prediction is applied in several areas of life, from the weather to the valuation of stocks in the stock market, this is defined as: "action and to predict" as "the words that manifest what is predicts "; in this sense, to predict something is "to proclaim by revelation, science or conjecture something that is to happen" [2].

Word prediction is one of the most commonly used methods to help people with motor disabilities in both personal communication and writing. The advantages derived from their use of prediction depend on the degree of motor disability, their linguistic ability, etc., and is mainly measured in the acceleration in writing, and in the reduction of effort required to write text. Of the various applications for people with motor disabilities, there are success stories such as predictive editors, language teaching systems, and translators. In the case of the editors have been developed in several languages, among them, the Spanish language [3].

One of the editors created consists of a single button which is used through different menus that are presented sequentially and the user must press the button to select the desired option. This way the user can write text. The prediction of text allows writing complete words with the same effort that would entail writing a letter [4].

In the case of language teaching systems, including Spanish as a second language, it opens the door to the application as a predictive engine for teaching language as a pre-spelling checker. The system can pre-correct the student's options at the time of writing and solve the question that arises if a word is written with g or j, b or v, s or z, etc. One of the first implementations of a predictive tool, in the case of translation, was described in the form of a simple complete-word system based on statistical models [5].

For over 20 years, word prediction has been an important technique for augmentative communication. Traditional systems have used word frequency lists to complete words that the user has already begun explaining. However, the most sophisticated prediction techniques based on the previous word or syntactic rules have appeared in recent years [6].

In N-gram word prediction methods, (Figure 1) the n-1 above words are used to predict the current (nth) word. The n-gram data is collected by counting the occurrence of each unique n-word sequence in a large body of text called the training text. For augmentative communication applications, n-gram techniques have been limited to unigram (n = 1) and bigrams (n = 2) prediction words, although trigram (n = 3) and upper N-gram orders are commonly used in other language-related fields such as voice recognition and machine translation [7].

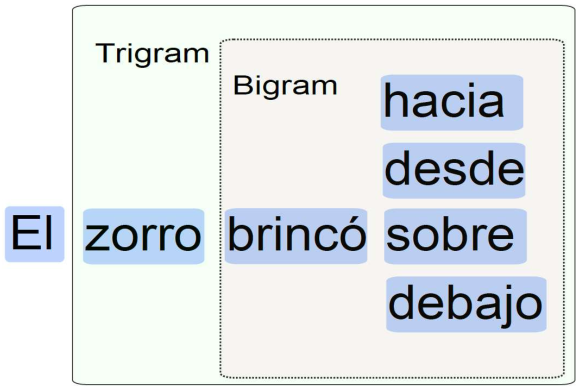


Figure 1. N-gram: Trigram and Bigram.

The N-grams have been used in various applications of artificial intelligence of which the following can be mentioned, mainly:

- In the segmentation of sentences based on Japanese language morphemes to detect which language structures are the most recurrent [8].
- In automatic segmentation of sentences in Korean to words and running spaces in word spacing errors when spelling [9].
- In the creation of a lexical analyzer based on the statistics of previous words and that served to know the regularity of certain expressions in the English language [10].

2. METHOD DESCRIPTION

The random process or stochastic process is denominated to a process or sequence of events that develops in the time in which the result in any stage contains some element that depends on the chance. The simplest case of a stochastic process in which the results depend on others occurs when the result in each stage depends only on the previous result and not on previous results. Such a process is called a Markov process or a Markov chain (a chain of events, each event bound to the preceding one) [11].

Given a sequence of random variables X_1; X_2; X_3; ...; such that the value of X_n is the state of the process at time *n*. If the conditional probability distribution of X_{n+1} in past states is a function of X*n* by itself,

so:

$$P\left(X_{n+1} = \frac{x_{n+1}}{X_n} = X_{n-1} = x_{n-1}, \dots X_2 = x_2, X_1 = X_1\right) = \tag{1}$$

$$P(X_{n+1} = \frac{x_{n+1}}{X_n} = x_n) \tag{2}$$

Where xi is the state of the process at time *i*. This identity is the so-called Markov property: the state at *t + 1* only depends on the state at *t* and not on the previous evolution of the system [11].

This analysis consisted of several stages. The first was to build a database of different users, by which it will be possible to obtain the characteristic patterns of programming of each one of them. The database consisted of a thousand files, part of which was used for training and another for validation.

The database is classified by users, and these, in turn, will have a certain number (variable depending on the case) of programming code files (*. ccp). Since it is already defined each user and their respective codes allow at the time of making the prediction to know if it is possible to give successful results or not.

In addition to the construction of the database, started with the development of a parser or lexicographic analyzer, which allows separating from each file where this content the code in various tokens, which establish the hierarchy on the trends of how the programmer (Figure 2). Tokens let you know when they are used and under what circumstances.

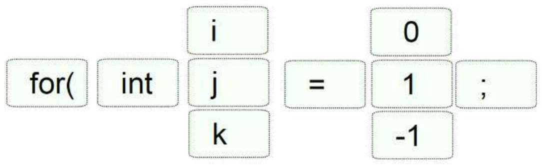


Figure 2. Characters spacing.

With all the above it is necessary to store this information to know the source of each token and in turn lay the basis of the statistics necessary to make the prediction in the future. Finally, this allows establishing the necessary bases to use the artificial intelligence method proposed an N-gram.

N-gram is an artificial intelligence method used mainly to process natural language, because of this, it will be a fundamental tool to be able to carry out the resolution of the problem.

Since the analysis pretends to be a reference in the statistical and implementation part of the artificial technique, it starts with the most basic structure of N-gram, the bigram, which consists of a square matrix of dimensions of NxN la which will allow to work in only two dimensions, to know if it is possible to carry out code prediction and how effective it can be.

To develop the bigram, a matrix is necessary, this one was based on the use of a programming structure pertaining to Standard Library Template (STL) of which two of its most common structures, vectors and maps were not used.

Returning the development of the lexicographic analyzer, the words or tokens obtained, were introduced in a map, together with a numerical value that allowed to know in which order is presented in the analyzed code and to be used later for the elaboration of the matrix.

Once the map was created and filled with the code information, the next step was to create an array, and for this, the second structure mentioned above, a vector, was used. This allowed for the flexibility needed by the matrix and the program, due to the nature of the analysis, this must be in a constant phase of learning, and therefore, present the capacity to remember and increase its memory.

For the bigram, we created a vector of vectors, which allowed us to create the matrix depending on the tokens introduced, and that this same genre then the statistical analysis to know which word will follow after which.

Once the NxN dimension matrix and the map filled with the tokens and their respective values in order of appearance were created, the values of the number of repetitions that were present in the programming code combinations were entered into the matrix (Table 1).

Table 1. Bigram Prediction Matrix.

	inicio	fin	int	i	;
inicio	0	0	1	0	0
fin	0	0	0	0	0
int	0	0	0	1	0
i	0	0	0	0	1
;	0	1	0	0	0

With the obtained matrix, this process was repeated again and again until having covered with all the codes designated for training.

Once the bigram was trained, a validation test was carried out to see if it is possible and with certain certainty to carry out the code prediction.

For this, a second application was developed, which will allow you to test with a second set of codes different from those used in training, and in turn, you will know that both are wrong with respect to that user.

Finally, all the analysis was repeated, but this time, using a Trigram, which carried with it the analysis of an array of NxNxN allowing to increase the memory and ability to perform the prediction to the program.

The database was obtained from a group of different programmers worldwide. For this purpose, the site http://www.planetsource-code.com was used as a resource, which contains a large collection of programming codes in different programming languages and languages.

The programming codes obtained are written in English language both in their comments and in their variables, so that allowed more standardization of information for this study.

The database consists of 1000 files including files of different sizes and with extension * .cpp and * .h that go inside each folder and which in turn have different sizes.

All these files are written and commented in English, but does not exclude that your application is useful for more languages.

This phase is the actual reading of the source program, which is usually in the form of a character stream and performs what is known as lexical analysis: it collects sequences of characters in significant units called tokens, which are the words of a natural language, like English.

In this way, you can imagine that a crawler performs a function similar to spelling.

Once the matrix and the map with their respective keys have been created, we begin with the elaboration of the bigram and the record of the combinations shown in the programming code, with which we finally have the state machine necessary to perform the code prediction.

The first word within the array is the combination of START_SYMBOL and the first word detected in the code, thus identifying that programming codes tend to form a certain pattern depending on the type of algorithm to be created and the programmer style.

Continuing, they are diagnosed which are the last and penultimate words of the list of tokens, with which, it is possible to know the dependence of said words between them. This way the entire list is traversed until the end of the source code.

After the bigram was built, the first tests were started, with training based on 80% of the database, and using the remaining 20%, to test.

Then the intermediate words of the programming code were taken into account so that the introduced word and the previous word were evaluated, and with the generated statistic the following prediction was made and it is counted as correct or incorrect as the case may be. At the end, after all the source code has been read for validation, it verifies if the code ends in a certain way that corresponds with the prediction of the bigram and the validation is done to know if the prediction was correct or not.

3. Results

The following tests were performed using the database consisting of 1000 different files, including files with .cpp extension and .h having an extensive vocabulary. The comparison will be carried out between two algorithms the bigram and the trigram and will highlight the advantages and disadvantages for the application in the prediction of code in C ++ language. In each test will be done with a group of training files, to have the greater reliability of the results and trying to establish a stochastic regime. The words presented are exclusive content of the codes stored in the database and the test words are extracted from them.

In this test, the bigram was trained with the total of files that are counted in the database (1000), arranged alphabetically and of different sizes, of extensions *. ccp and * .h, of which they are omitted comments and respects everything relevant in code, such as spaces, line breaks, tabulation, etc. This in order to know the programming style of the programmer who owns the programming codes. This test is intended to test the algorithm with over 196 thousand tokens, of which 6465 are unique.

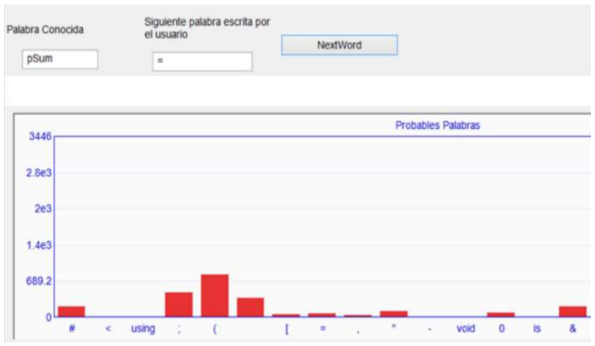
The files needed for validation are part of the database and are also present in the algorithm training, which is expected to have a low error rate and to be able to prove that the algorithm is trained.

Once the test was run simulations were made of which would be the possible words to use through the prediction and in turn be able to contrast this with the written by the author of the codes.

Only tokens that have a higher probability of being considered for the type of token entered will be shown.

From the validation files, there are a little more than 11 thousand tokens of which the test will be taken by taking some random to establish the efficiency of the algorithm and knowing well, if possible how well it is to make a prediction.

In this test, we can conclude (as shown in Figure 3) that the program has an efficiency of more than 90% because in 100 tokens taken at random to perform test was only wrong in 7 because of both the group of training codes as validation are different, all done with both n-grams.


Palabra Conocida: pSum

Siguiente palabra escrita por el usuario: =

Figure 3. Token Prediction

It should be mentioned that the other tokens are not discarded as they are the symbol less than (<) because they also have an effect on what was shown in the training, this test can be considered valid because it can be seen that in most the cases the program shows a group of tokens with high probability of being a possible option for the user.

4. Conclusions

In this paper, we have presented the comparison of two n-grams, bigram, and trigram, for the purposes of the analysis, it was possible to conclude that the bigram has sufficient capacity and is less demanding in terms of memory and processor than the trigram and that despite having a shorter memory effect is quite efficient. The development in other languages could be possible only with slight adaptations of the code and with a new database to carry out the training and the validation.

It can also be thought that in the future this work will serve as a basis for tests that show their effectiveness against other methods such as Artificial Neural Networks.

Also among the discovered tokens is intended to make an application that when integrated into an IDE is more comfortable and agile the experience of programming mainly for beginning users.

Acknowledgements

The authors thank the University of Guanajuato Irapuato-Salamanca campus and the Technological Institute of Purísima del Rincon for the facilities provided.

REFERENCES

[1] Charniak, E., Riesbeck, C. K., McDermott, D. V., & Meehan, J. R. (2014) *Artificial intelligence programming*, Psychology Press.

[2] Brodie, M. L., Mylopoulos, J., & Schmidt, J. W. (Eds.). (2012) *On conceptual modelling: Perspectives from artificial intelligence, databases, and programming languages*, Springer Science & Business Media.

[3] Palazuelos, S. E., Rodrigo, J. L., Godino, J. I., Aliaga, F., Martín, J. L., & Aguilera, S. (1999)"Predicción de palabras en castellano," *Procesamiento del lenguaje natural*, Vol. 25.

[4] Foster, G., Langlais, P., & Lapalme, G. (2002) "User-friendly text prediction for translators", In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing,* Vol. 10, pp148-155.

[5] Nobesawa, S., Tsutsumi, J., Jiang, S. D., Sano, T., Sato, K., & Nakanishi, M. (1996) "Segmenting Sentences into Linky Strings using D-bigram statistics", In *Proceedings of the 16th conference on Computational linguistics,* Vol. 2, pp586-591.

[6] Kang, S. S., & Woo, C. W. (2001)"Automatic Segmentation of Words using Syllable Bigram Statistics", In *NLPRS,* pp729-732.

[7] Collins, M. J. (1996) "A new statistical parser based on bigram lexical dependencies", In *Proceedings of the 34th annual meeting on Association for Computational Linguistics* pp184-191.

[8] Lesher, G. W., Moulton, B. J., &Higginbotham, D. J. (1999)"Effects of ngram order and training text size on word prediction", In *Proceedings of the RESNA'99 Annual Conference,* pp52-54.

[9] Aldezábal, I., Alegria, I., Ezeiza, N., &Urizar, R. (1996)"Del analizador morfológico al etiquetador/lematizador: unidades léxicas complejas y desambiguación", *Procesamiento del lenguaje natural*, Vol.19.

[10] Gutiérrez, J. J., Escalona, M. J., Villadiego, D., & Mejías, M. (2005) "Comparativa de herramientas para la enseñanza de lenguajes relacionales", *Actas de las XI Jornadas de Enseñanza Universitaria de la Informática*, Vol.11, pp297-304.

[11] Arya, J. C., &Lardner, R. (2002) *Matemáticas aplicadas a la administración y a la economía*, Pearson Educación.

Authors

Sergio Rodríguez Miranda

PhD. Student. In Ciatec (2014-2018) on Industrial Eng. Bs. On Mechatronics. Assc. Prof.In ITSPR.

Mexico · Guanajuato

Photo

Luis Fernando Villanueva Jiménez

PhD. Student. In Ciatec (2014-2018) on Industrial Enginiering. Bs. On Industrial Engineering.Assc. Prof.In ITSPR.

Mexico · Guanajuato

Sergio Eduardo Ledesma Orozco

PhD. In Electrical Engineering at Stevens Institute of Technology, NJ. USA

Research experience

Jul 2004–present

Professor (Full) Universidad de Guanajuato Division of Engineering,

Mexico · Guanajuato