Iraq University College Journal of Engineering and Applied Sciences

Comparison of Markov Chain and traditional method for Probabilities of Arabic Language letters based on holly Qur'an

Hamid A. Al-Asadi[1,a], **and Hayder K. Alhamadi**[1,b*]

1 Communications Engineering Department, Iraq University College, Basrah - Iraq
2 Alzahraa Medical College, Basrah - Iraq
E-mail: [a]865.hamid@gmail.com, [b,*]haidarfmg@email.com

Abstract. The Arabic language has very complex structures and different behavior, where instead of moving a letter to another letter like English language, it moves to a diacritic. Data analysis studies of a language are essential in numerous fields of knowledge, like Education. linguistics, Computers and Communications.The aim of this paper is to study the probabilities of letters in the Arabic frequencies statistics based on holly Quran sharif words sample Then deduced and compared with corresponding values for Markov chains.

Keywords: Diacritics; Python; Dictionary; Markov chain; Arabic text

Iraq University College Journal of Engineering and Applied Sciences Volume 1, Issue 2
Received 03/09/2021
Accepted 03/11/2021
Published 20/12/2021
https://magazine.iuc.edu.iq/journal/Online at
DOI:yy.yyyy/ IUCJEAS.2021.xx.x.xx

1. Introduction

In the field of grand probabilities, there are many tools for calculating probabilities and predictions; Specifically, Markov chain theory is used to predict certain electoral outcomes. In order to make predictions about a politician's career, a Markov chain and related tools must be created based on the scenario and probabilities. A Markov chain is a mathematical system that experiences transitions from one state at time t to another at time t+1 according to certain probability rules. The hallmark of a Markov chain is that no matter how a process arrives at its current state, the possible future states are constant [11].

A Markov chain can show scenarios that include probabilities more clearly because it shows the transition between states. Using Markov chain theory, more and more realistic election results can be expected [12].

Natural Language Processing (NLP) applications that utilize statistical approach, has been increased in recent years. One of the most important models of machine learning used for the purpose of processing natural language is Hidden Markov Model (HMM). Markov Model is a probabilistic model that are considered as sequence classifier such as letters classifier; it calculates the probability of label sequence and chooses the best sequence according to the best possible labels probability distributions. Moreover, Hidden Markov Model is a model that contains a set of state and transitions where transition from one state to another state is determined according to certain input. Each transition contains a value or weight that is determined according to certain probability distribution. Therefore, if certain input causes transmission from state x to state y then the overall weight will be augmented by the weight w that is the value of transition or transition probability between state x and state y. The probability distribution of a certain transition determines the observation or outcome of a certain state. However, Hidden Markov model is called hidden since the states are not visible and only outcomes can be seen. In our case the input is a sequence of words or letters, so the sequence of words will determine the sequence of states; this sequence represents a chain called Markov chain.

Hidden Markov Model was used in many applications of statistical NLP such as morphological analysis, part of speech tagging (PoST) and text classification. This research provides a comparative study between different applications using Hidden Markov Model in statistical language processing of Arabic language

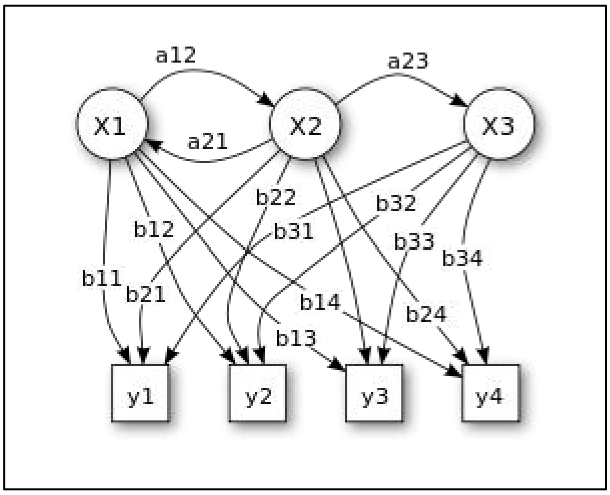


Fig. 1. Hidden Markov model

The Arabic language has very complex structures and different behavior, where instead of moving from a letter to another letter like English language, it moves to a diacritic as shown in the figure below:

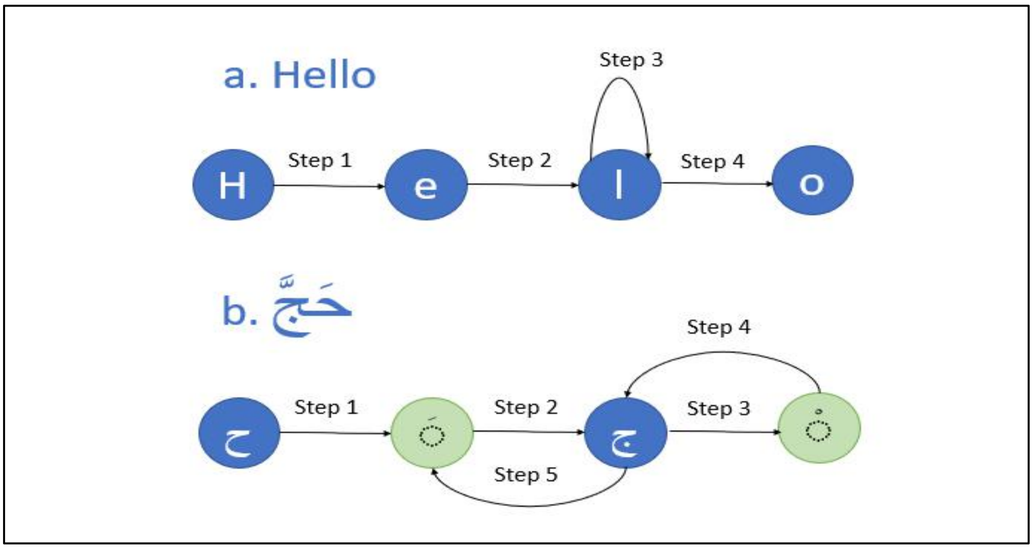


Fig. 2. (a) English language behavior (b) Arabic language behavior

So instead of dealing with diacritic as a small mark over or below a letter we're going to deal with it as a letter that have 50% percent of probability, so how can we get text have 50% diacritics based on Holly Quran. Also, we're going to show the different between the traditional method to get probabilities and the Markov method and how we will analysis this language. Also, how the researcher analyzed it using Hidden Markov Model (HMM)

It so complex to understand the behavior of letters or diacritics, for example when the "ج" go to "◌َّ" that's actually mean double letter of "ج" the first go to "◌ْ" and the second go to "◌َ" (" حَجْجَ ") and that just a little bit example, Even when I am an Arabian and native speaker but it is still difficult to understand the terms, conditions and the syntax of this language, so the best choice that we can rely on is the Holy Quran where each word and diacritic in right place, as much as we know that Arabic language has 28 letters and 4 main diacritics but actually we have more specially in Holly Quran but it still has low probability so I will focus on the letters and diacritics that have high probability but i will take it into consideration of my calculation because we need every small number to get high accuracy.

Actually, the aim of this paper is not to calculate the transition matrix or probabilities, because that already done by more researches, so the aim of this paper is to calculate this parameter in high accuracy using different tools that should help. As we see in Figure below, where whatever was the method, the result still have low accuracy, so Holly Qur'an may be a good choice to based on and with some help from computer and data science tools we're going to get high accuracy, we need this accuracy because it will be based to other applications so if the base have low accuracy we're not going to get the right result.

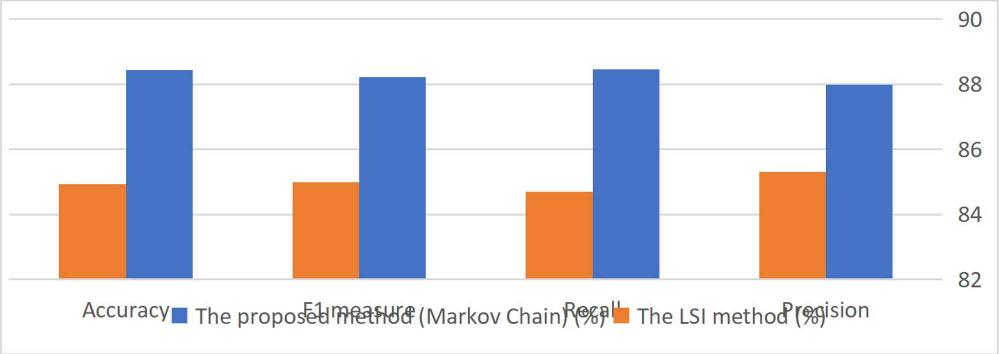


Fig. 3. Performance comparison of the Markov chain and the LSI method [13].

2. The Traditional method

When we talked about probability, we referred to Markov Chain but what about the basics of probability. When we have coin the probability of each side is 0.5, and when we have the word "yes" that mean every letter have probability 1/3 and if we talk about the male’s genes, he carries an X and another Y chromosome and writes XY, and arranged from two X chromosomes in the female and writes XX, so the probability of baby is [x, x, x, y] and that mean probability of male baby is 0.25 and probability of female baby is 0.75, that is the basic of probability or traditional method.

I will use Python language (as much as we know it is a robust language in data science and analysis) to analysis and calculate a text file actually it is a txt copy of Holly Quran. First, I will count how many frequency every single letter or diacritics and put it in an array then find the size of this text file in unit of character, to minimize the processing time I put the value of size in the end of the array then each value of this array will divided by the value of whole size to get occurs probability of each item in this array.

```
f = open("fmg.txt", encoding="utf-8", errors="strict")
x = f.read()
originallist = list()
originaldict = dict()
for i in x:
        originallist.append(i)
        originaldict[i] = 0
for i in range(len(originallist)):
        originaldict[originallist[i]] += 1
for i in originaldict:
          originaldict[i] = round(originaldict[i]/len(originallist),20)
```

In this code I round the number to 20 placements after point to get high accuracy. As we know the Arabic language is highly redundant and we can use this redundant to analysis the language as we see in the table below the word "في" has a high probability 3.9% actually it is a very high probability, later we will get result that improve this table.

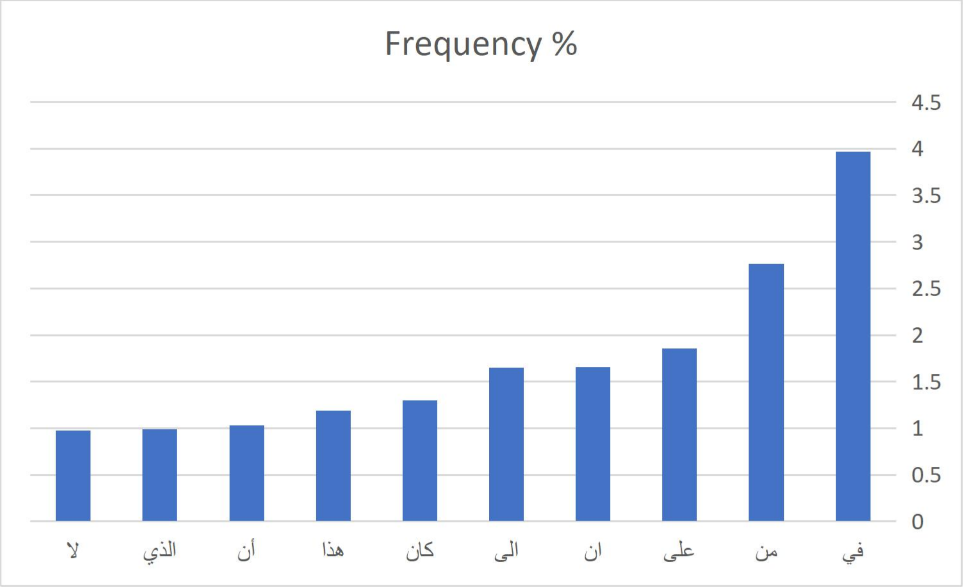


Fig. 4. (a) A list of the 10 most frequent Arabic words [12]

3. Markov Chains:

3.1 Transition matrix

If one Markov chain has state space S = {1, 2, . . . n}, the probability of this process in state j for one observation after being in state i in a previous observation, is denoted by Pij. This Pij is known as the transition probability from state i to state j. A matrix with a transition probability from state i to state j is known as the transition matrix of the Markov chain. Subsequently, the transition matrix is denoted with P.

To predict the probability of an action repeating over time, then you need to use a transition matrix. A transition matrix consists of a square matrix that gives the probabilities of different states going from one to another. With this transition matrix, we can perform determine trends and make predications, what we want actually are the letter history and the letter future (where to go). The general form of transition matrix is as follows:

$$\text{P} = \begin{bmatrix} P11 & \cdots & P1n \\ \vdots & \ddots & \vdots \\ Pn1 & \cdots & Pnn \end{bmatrix} \qquad (1)$$

Now I will create a new 2-dimention dictionary (matrix) and calculate how many times the item goes to every item:

```
dictionary2d = dict()
for i in originallist:
    dictionary2d[i][originallist[originallist.index(i)+1]] += 1
```

$$\text{P} = \begin{bmatrix} N11 & \cdots & N1n \\ \vdots & \ddots & \vdots \\ Nn1 & \cdots & Nnn \end{bmatrix} \qquad (2)$$

Then, I add a temporary item (S) at the end of each row of this matrix where S is the summation of the values of the same row, Mathematically, that characteristic can be written as follows:

$$S_i = N_{i1} + N_{i2} + \ldots.. + N_{in} \qquad (3)$$

```
for i in list(dictionary2d.keys()):
    sumall = 0
    for j in list(dictionary2d[i]):
           sumall += dictionary2d[i][j]
     dictionary2dProb[i]['sumall'] = sumall
```

$$\text{P} = \begin{bmatrix} N11 & \cdots & N1n & S1 \\ \vdots & \ddots & \vdots & \\ Nn1 & \cdots & Nnn & Sn \end{bmatrix} \qquad (4)$$

The state vector X(t) for one Markov chain observation with state space S = {1, 2, . . . k} is defined as the vector of column x where the i component, namely xi, is the probability of state i at time t. The column vector can be formulated as:

$$\text{x} = \begin{bmatrix} x1 \\ x2 \\ . \\ . \\ . \\ xk \end{bmatrix} \qquad (5)$$

According to theorem by Anton and Rorres (1987), if P is the Markov chain transition matrix and x(n) is the state vector at observation n, it makes:

$$x^{(n+1)} = Px^{(n)} \quad (6)$$

From Eq. (6), it is known that:

$$x^{(n)} = Px^{(n-1)} = P^2\, x^{(n-2)} = P^n\, x^{(0)} \quad (7)$$

In other words, Eq. (7) verifies that the previous state vector x(0) and transition matrix P reveal the value of state vector x(n).

Finally, we divide every item of row i by the summation of the same row and round it to 20 placements after point and after that we will delete the summation item to get the transition matrix as shown in Eq. (1) and the piece of code will be as follow:

```
for i in list(dictionary2dProb.keys()):
   for j in list(dictionary2dProb[i]):
       if(j != 'sumall'):

            dictionary2dProb[i][j]=round(dictionary2dProb[i][j]
            /dictionary2dProb[i]['sumall'],20)
            del dictionary2dProb[i]['sumall']
```

3.2 Markov Chains

Because it is complex to draw all the transition Matrix I will take one letter and draw it as follow in the figure below:

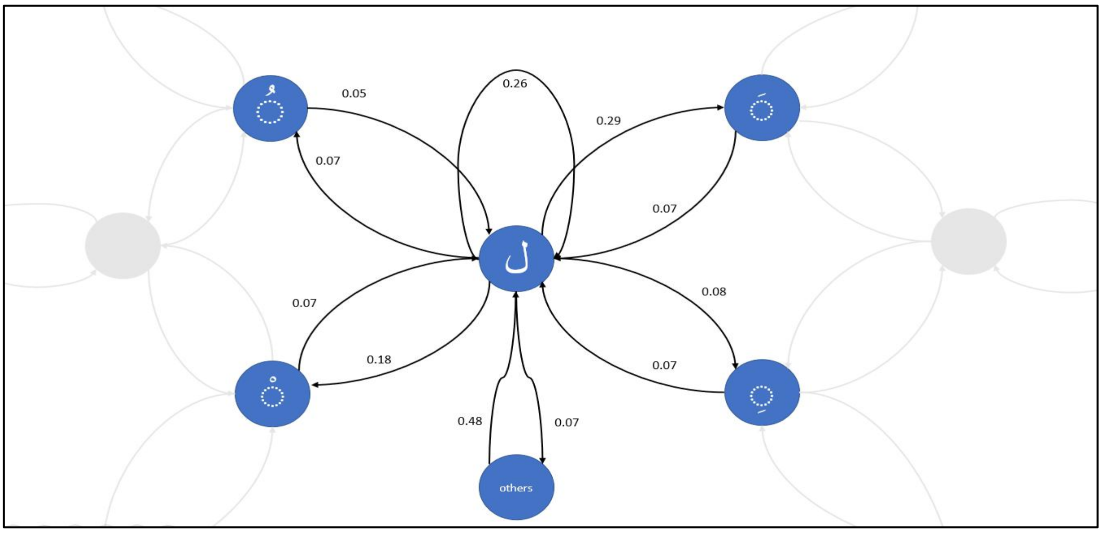


Fig. 5. Markov Chain for "ل"

As we see in Fig. 1 above it a very complex network of chains to calculated by hand so the solution is the same tool (Python).

From Eq. (1), Pij verifies the transition probability from state i at time t to state j at time t+1. In addition, the Markov chain transition matrix above has the characteristic that all entries on one line equal 1. Mathematically, that characteristic can be written as follows:

$$P_{i1} + P_{i2} + P_{i3} + P_{i4} + \ldots\ldots + P_{in} = 1 \quad (8)$$

And the probability of an item I calculated by summation of probabilities of each item multiply by its transition probability. Mathematically, that characteristic can be written as follows:

$$P_i = P_{1i} * P_1 + P_{2i} * P_2 + P_{3i} * P_3 + ... + P_{ni} * P_n \qquad (9)$$

Based on Eq. (8) and also Eq. (9) we can find all the probabilities that we need with some help from computer processing as much as you know it is so complex to solved by hand.

4. Results

We talked in section two about probabilities by the traditional way and how can find it using Python, So the result of this section as follow in the figure bellow:

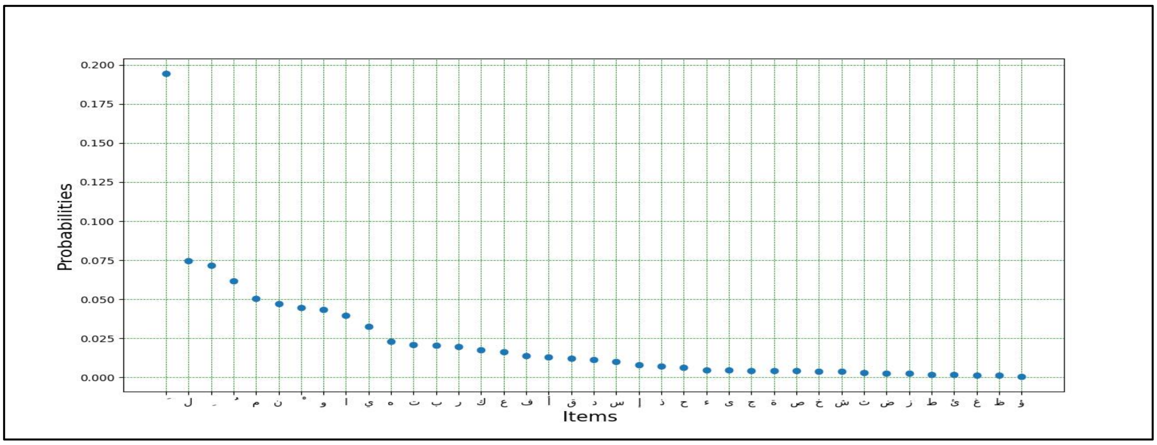
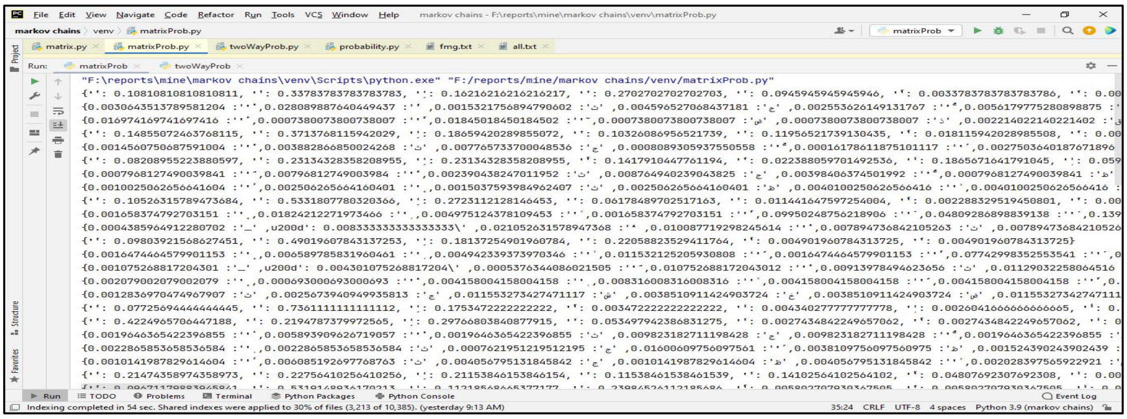


Fig. 6. The probability chart of the array

One of the most important features of Python is the graph of data as we see in Fig. (1) in addition to the four main diacritics, we have some letters that have high probability and we see these letters a lot specially in Holly Quran and who didn't hear "ألم". We see the gap between Fatha and the other letters and diacritics, that an approve to "Musnad Al-Imam Ahmed". And also, that is approve the table (1) where we will find of the letters (that have probability greater than 0.025) in the words of table (1).[8]

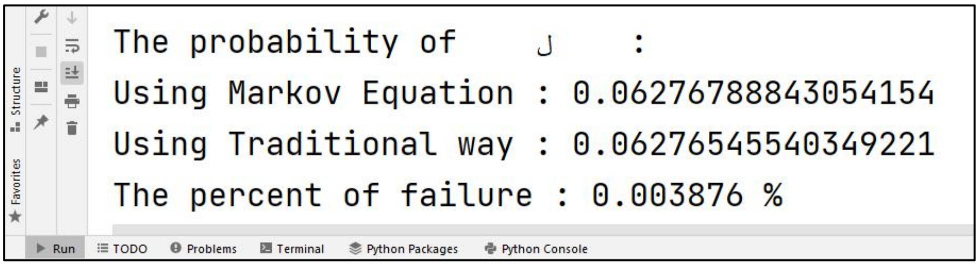
After that we talked about calculation the transition matrix after delete the item of summation of each row, but I used 20 placements after point to each number and matrix have more than 30 items in each row so it will be huge data, so I will show part of this data in the figure bellow:


File Edit View Navigate Code Refactor Run Tools VCS Window Help
markov chains - F:\reports\mine\markov chains\venv\matrixProb.py
markov chains 〉 venv 〉 matrixProb.py
matrixProb
matrix.py
matrixProb.py
twoWayProb.py
probability.py
fmg.txt
all.txt
Project
Run:
matrixProb
twoWayProb
Structure
Favorites
Run
TODO
Problems
Terminal
Python Packages
Python Console
Event Log
35:24 CRLF UTF-8 4 spaces Python 3.9 (markov chains)

```
"F:\reports\mine\markov chains\venv\Scripts\python.exe" "F:/reports/mine/markov chains/venv/matrixProb.py"
{'َ': 0.10810810810810811, 'ُ': 0.33783783783783783, 'ِ': 0.16216216216216217, 'ْ': 0.2702702702702703, 'ّ': 0.0945945945945946, 'ً': 0.0033783783783783786, 'ٌ': 0.00
{0.0030643513789581204 :'ٍ',0.028089887640449437 :'ـ' ,0.0015321756894790602 :'ث' ,0.004596527068437181 :'ج' ,0.002553626149131767 :'ء',0.0056179775280898875 :'
{0.016974169741697416 :'ْ',0.0007380073800738007 :'ً',0.01845018450184502 :'~',0.0007380073800738007 :'ض' ,0.0007380073800738007 :'ذ' ,0.002214022140221402 :'ق
{'َ': 0.14855072463768115, 'ُ': 0.371376811594202 9, 'ِ': 0.1865942028985507 2, 'ْ': 0.10326086956521739, 'ّ': 0.11956521739130435, 'ً': 0.018115942028985508, 'ٌ': 0.00
{0.0014560750687591004 :'ّ',0.003882866850024268 :'ث' ,0.0077657337000048536 :'ج' ,0.0008089305937550558 :'ء',0.00016178611875101117 :'ً',0.0027503640187671896
{'َ': 0.0820895522388059 7, 'ُ': 0.23134328358208955, 'ِ': 0.23134328358208955, 'ً': 0.1417910447761194, 'ٌ': 0.022388059701492536, 'ٍ': 0.1865671641791045, 'ِ': 0.059
{0.0007968127490039841 :'ٓ',0.00796812749003984 :'ّ',0.0023904382470 11952 :'ث' ,0.008764940239043825 :'ج' ,0.003984063745019 92 :'ء',0.0007968127490039841 :'ظ'
{0.0010025062656641604 :'ٍ',0.002506265664160401 :'ِ',0.0015037593984962407 :'ث' ,0.002506265664160401 :'ظ' ,0.004010025062656 6416 :'ٔ',0.004010025062656 6416 :
{'َ': 0.10526315789473684, 'ُ': 0.5331807780320366, 'ِ': 0.272311212814 6453, 'ْ': 0.06178489702517163, 'ّ': 0.011441647597254004, 'ً': 0.002288329519450801, 'ٌ': 0.00
{0.0016583747927 03151 :'ِ',0.0182421227197 3466 :'ِ',0.004975124378109453 :'ٔ',0.0016583747927 03151 :'ً',0.0995024875621 8906 :'ٓ',0.04809286898839138 :'ٓ',0.139
{0.00043859649122 80702 :'_' ,u200d': 0.0083333333333333 33\' ,0.0210526315789473 68 :'ـ ,0.010087719298245614 :'ّ',0.007894736842105263 :'ث' ,0.00789473684210526
{'َ': 0.09803921568627451, 'ُ': 0.49019607843137253, 'ِ': 0.18137254901960784, 'ْ': 0.22058823529411764, 'ً': 0.004901960784313725, 'ٌ': 0.004901960784313725}
{0.0016474464579901153 :'ِ',0.006589785831960461 :'ِ',0.004942339373970346 :'ٔ',0.011532125205930808 :'ٔ',0.0016474464579901153 :'ً',0.07742998352553541 :'ٓ',0
{0.001075268817204301 :'_' ,u200d': 0.004301075268817204\' ,0.0005376344086021505 :'ٓ',0.010752688172043012 :'ّ',0.009139784946236 56 :'ث' ,0.0112903225806 4516
{0.002079002079002079 :'ِ',0.000693000693000693 :'ّ',0.004158004158004158 :'ِ',0.008316008316008316 :'ٔ',0.004158004158004158 :'ٔ',0.004158004158004158 :'ً',0.
{0.0012836970474967907 :'ث' ,0.0025673940949935813 :'ج' ,0.011553273427471117 :'ش' ,0.0038510911424903724 :'خ' ,0.0038510911424903724 :'ض' ,0.01155327342747111
{'َ': 0.07725694444444445, 'ُ': 0.7361111111111112, 'ِ': 0.1753472222222222, 'ً': 0.003472222222222222, 'ٍ': 0.004340277777777778, 'ِ': 0.0026041666666666665, 'ٕ': 0.
{'َ': 0.4224965706447188, 'ُ': 0.2194787379972565, 'ِ': 0.29766803840877915, 'ْ': 0.053497942386831275, 'ّ': 0.0027434842249657062, 'ً': 0.0027434842249657062, 'ٍ': 0
{0.0019646365422396855 :'ٍ',0.005893909626719057 :'ّ',0.0019646365422396855 :'ث' ,0.009823182711198428 :'ج' ,0.009823182711198428 :'ء',0.0019646365422396855 :'
{0.0022865853658536584 :'ِ',0.0022865853658536584 :'ث' ,0.0007621951219512195 :'ج' ,0.01600609756097561 :'ً',0.0038109756097560975 :'ظ' ,0.001524390243902439 :
{0.0010141987829614604 :'ٍ',0.006085192697768763 :'ث' ,0.004056795131845842 :'ج' ,0.0010141987829614604 :'ظ' ,0.004056795131845842 :'ٔ',0.002028397565922921 :'
{'َ': 0.21474358974358973, 'ُ': 0.22756410256410256, 'ِ': 0.21153846153846154, 'ْ': 0.11538461538461539, 'ّ': 0.14102564102564102, 'ً': 0.04807692307692308, 'ٌ': 0.00
{'َ': 0.09671179883945841, 'ُ': 0.5319148936170213, 'ِ': 0.11218568665377177, 'ْ': 0.23984526112185686, 'ً': 0.005802707930367505, 'ٌ': 0.005802707930367505, 'ٍ': 0.0
Indexing completed in 54 sec. Shared indexes were applied to 30% of files (3,213 of 10,385). (yesterday 9:13 AM)
```

Fig. 7. The Transition Matrix

Now the most important part and the final result where as we said in section (3) we will calculate the value of probability for "ل" for example compared to the value of probability of the same letter using traditional method and also, we're going to calculate the percent of error to see the accuracy, and the result shown in the figure bellow:



```
The probability of    ل    :
Using Markov Equation : 0.06276788843054154
Using Traditional way : 0.06276545540349221
The percent of failure : 0.003876 %
```

Fig. 7. Probability of "ل" using Markov Equation compared to traditional way

4. Conclusion

It takes more time but finally, it is as I wanted and the result is with high accuracy. Its huge data but I tried some examples to make sure that everything is ok. From figure (4) we see that we have high accuracy in our calculation and just 0.004% is an error ratio, so why we make this complex and we can get the probability from the first method (traditional method), actually in the first method we get only the probability but using Markov chains we get probability and the transition matrix. this helps us to understand the behavior of the letters and diacritics, it's so important to know the future of letters (where to go) and write new and fast algorithms about autocomplete text where the current algorithms depend on user experience but user experience sometimes give us wrong results and always give us result without correct diacritics because those who specialize in language are the minority and those who determine user experience are the majority who have little experience with grammars and terms.

References

[1] G. A. Miller, Language and Speech. San Francisco: Freeman and Co., USA, 1981.

[2] A. A. Abuseeni, “Computational method for accurate classification of Arabic texts based on Arabic phonetic transcription”, GLOBAL JOURNAL FOR RESEARCH ANALYSIS. Vol. 4,issue -3, pp. 9- 16, March 2015.

[3] Y. A. Rozanov, Probability Theory: A Concise Course (Revised English Edition Translated & Edited by R. A. Silverman), Dover Publications, New York, USA, 1972.

[4] Nagpal, Abhinav & Gabrani, Goldie. Python for Data Analytics, Scientific and Technical Applications. 140-145. 10.1109/AICAI.2019.8701341, 2019.

[5] King Abdu1aziz Public Library (Ed). Proceedings of Symposium on Using Arabic Language in Information Technology (8-12 Dhu Al Qadah 1412 AH. / 10-14 May 1992), (in Arabic), King Abdu1aziz Public Library Press - Authentic Works Series (4). Riyadh. 1993.

[6] M. Gr. Voskoglou, An application of Markov Chains to Decision Making, Studia Kupieckie (University of Lodz), 6, 69-76, 2000.

[7] S. M. Abusini, “ Morphological and phonetic reading in Arabic language structure”, Zarka J. Res. Stud. 7:1, 2, 2005.

[8] Mohammed Aabed, Sameh Awaideh, Abdul-Rahman Elshafei, and Adnan Gutub, “Arabic Diacritics Based Steganography”, IEEE International Conference on Signal Processing and Communications (ICSPC 2007), Pages: 756-759, Dubai, UAE, 24-27 November 2007.

[9] Porter, Theodore M.. "probability and statistics". *Encyclopedia Britannica*, 3 Feb. 2020

[10] Yang, Xinye. Markov Chain and Its Applications. 10.13140/RG.2.2.12289.61287, 2020.

[11] Grinstead & Snell. Introduction to Probability. American Mathematical Society. Providence, 2003.

[12] I. A. AI-Kadi, " Study of Information-theoretic Properties of Arabic Based on Word Entropy and Zipf's Law", King Saud University, P.O. Box 800. Riyadh 11421, Saudi Arabia, , 01 January 1996.

[13] F. S. Al-Anzi, D. AbuZeina, " Beyond vector space model for hierarchical Arabic text classification: A Markov chain approach", Department of Computer Engineering, Kuwait University, Kuwait, 11 October 2017.

Hamid A. Al-Asadi was born in Iraq. He received the B.Sc and M.S. degrees in electrical engineering and communication engineering from Basra University, Basra, Iraq, in 1987 and 1994, respectively, and the Ph.D. degree from the University Putra Malaysia in Communication Network Engineering in 2011. From 1995-2018, he was a faculty member in the Department of Computer science, Basra University. In 2014, he joined the Basra University as a Full Professor. Since November 2018 he has been head of the Department of communication engineering in the Iraq University College, Iraq. His research interests include optical communications, optical fiber, information theory, Wireless Network, Sensor Network, Fuzzy Logic and Neural Networks, Swarm Intelligence, computer engineering, and Artificial intelligence. He is member of scientific and reviewing committees of many journals and international conferences in the domains of Computer and communications engineering.

Hayder K. Alhamadi was born in Iraq. He received the B.Sc degree in computer engineering and from Basra University, Basra, Iraq, in 2019. In 2020, he joined the Basra University. He is a Software Engineer now. His research interests include Data Science, Data Analysis, information theory, Database management, Server management, Fuzzy Logic and Neural Networks, computer engineering, and Artificial intelligence.

DOI: yy.yyyy/IUCJEAS., 2021.vol 1., issue.2

مجلة كلية العراق الجامعة للهندسة والعلوم التطبيقية

مقارنة بين سلسلة ماركوف والطريقة التقليدية لاحتمالات حروف اللغة العربية على أساس القرآن الكريم

حامد علي عبد الاسدي [1،أ] , حيدر الحمدي [2، ب] .

1 كلية العراق الجامعة – قسم هندسة الاتصالات – البصرة – العراق

2 جامعة البصرة – كلية طب الزهراء – البصرة – العراق

البريد الالكتروني : 865.hamid@gmail.com [أ] , haidarfmg@gmail.com [ب]

الملخص . للغة العربية هياكل معقدة للغاية وسلوك مختلف ، حيث بدلاً من انتقال الحرف إلى حرف آخر مثل اللغة الإنجليزية ، فإنه ينتقل إلى التشكيل(الحركة) والتي تعد اساسية في تفسير وتمييز الكلمات والنصوص. تعد دراسة تحليل بيانات اللغة وخصائصها ضرورية في العديد من مجالات المعرفة ، مثل تعليم اللغات والحاسوب والاتصالات. يرمي هذا البحث الى دراسة احتمالات الحروف في إحصائيات الترددات العربية بناءً على عينة كلمات القرآن الكريم ثم استنتاجها ومقارنتها مع القيم المقابلة لسلاسل ماركوف. ومنها دراسة مقارنة بين سلسلة ماركوف والطريقة التقليدية لاحتمالات حروف اللغة العربية وحساب مصفوفة الانتقال التي ستكون ذات اهمية قصوى في دراستنا, بالاضافة الى تسليط الضوء على نموذج ماركوف المخفي.