Information Systems and Signal Processing Journal (2020) 5: 1-4
Clausius Scientific Press, Canada
DOI: 10.23977/isspj.2020.51001
ISSN 2371-9427

The First Azeri (Azerbaijani) Language Next Word Predictor

Ali Pourmohammad
Process Automation Engineering Department
Baku Higher Oil School
Baku, Azerbaijan
pourmohammad@bhos.edu.az

Mensur Gulami, Javid Mahmudov,
Yusif Aliyev, Rovshan Akberov, Anar Sultani
Department of Computer Science
Khazar University
Baku, Azerbaijan

***Abstract*— Azeri (Azerbaijani) language is one of the more than 50 Turkic languages which it is a little studied language in terms of using the modern signal processing algorithms. This paper tackles the problem of Hidden Markov Models (HMMs) based next word prediction for this language based on Natural Language Processing (NLP) principles using Python high-level programming language. The software is included a small Azeri vocabulary database, the various Python libraries, a HMM model and a Web based interface. In this research, the database was constructed by a predictor parser which it was implemented for the first time for Azeri language. The database was concluded by the most general Azeri language words to introduce HMMs based generated word pairs. The Model was trained by 90% of the database, hence, predicting the next 5 words on the test data resulted 54% accuracy.**

Keywords*— Azeri (Azerbaijani) Language; Next Word Predictor; Hidden Markov Model (HMM), Natural Language Processing (NLP)

I. INTRODUCTION

Azeri (Azerbaijani) language is one of the more than 50 Turkic languages [1] which it is a little studied language in terms of using modern signal processing algorithms and creation of modern language technology applications [2]. despite a huge number of researches on the other languages since the 80th years of the last century, Azeri language is a little investigated language, where all those researches studied applying Automatic Speech Recognition (ASR), Text-To-Speech (TTS) or Authorship Recognition (AR) algorithms on this language as "Dilmanc" project [2-6]. For the first time, the next word prediction for Azeri language has been mentioned in this research. Nowadays, one of the most important real-time social media's necessaries is electronically conversation and communication. Reducing the time consumption for typing in the electronically communications by means of the next word prediction, would be very helpful for day to day usage. Hence, during the last decade, one of the highly discussed topics in Natural Language Processing research domain was the next word prediction for typing in the electronically communications [7].

Recurrent Neural Networks (RNNs) as Long Short-Term Memory Network (LSTM) model and Hidden Markov Models (HMMs) are used to perform prediction of the next word on a text sequence [7-11]. When the training data is not enough, and it is limited, HMMs can be used competitively on action modeling tasks in comparison with the discriminatively trained RNNs. HMMs should be used for the short-term action memory (early learning) cases or when quick reaction is required and RNNs should be used when enough data is available (long-term action memory). Also, using HMMs, the time complexity will be decreased compared to using RNNs as a LSTM network [12]. HMMs are easier to implementation, and still quite powerful. Therefore, it was decided to use HMMs in this research

This paper tackled the problem of HMMs based next word prediction for Azeri language, based on NLP principles using Python programming language. The software was included a small Azeri vocabulary database where the amount of it was about 4 MB, the various Python libraries, a HMM model and a Web based interface. In this research, the database was constructed by a predictor parser which it was implemented for the first time for Azeri language. The database was concluded by the most general Azeri language words to introduce HMMs based generated word pairs.

The structure of this paper is as follows. After the introduction section, it will be shortly reviewed HMMs and will be discussed the training on the model. Then, it will be explained the collection of the database issue. After introducing all parts of the software, some experimental results will be discussed. Finally, conclusions will be made in the last section.

II. HMMS AND THE TRAINING ON THE MODEL

The task of this research was to have a model which can predict the next word based on N previously written words. In this case, we set N to 2. The main reason to choose that specific number was to reduce the complexity of the model, since we were not dealing with enterprise level application and to reduce the overall computation cost. This very specific setting enabled us to scale once we found the way to generalize. To do this specific task we had several methods. The first one was to build a recurrent neural network as LSTM, to accomplish this task. Currently, quite a lot of enhancements have been spotted in this area. However, preprocessing and training an LSTM network would, first, be quite a challenging task with limited computation power we had. We could have used online services such as Google Colab, but these systems have all sorts of consistency problems. The second reason not to choose a neural network was that it would be an overkill for a small task. The next solution was to use Hidden Markov Models (HMM). We decided upon HMM, because it was easier to implement, and yet quite powerful.

HMM is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobservable states. HMM can be represented as the simplest dynamic Bayesian network. The mathematics behind the HMM were developed by L. E. Baum and coworkers. HMM works on previously observed states and tries to find a probabilistic relationship between given states [13].

In the design phase, we defined 3 tiers for our application. The first one was the Preprocessing tier (Fig. 1). It deals with raw text data. Main functionalities of Preprocessing tier included reading raw text from file, removing all the special symbols like dash, quotes, exclamations, questions, etc. We removed all the extra spaces, URLs, email addresses, numbers, braces, parenthesis, etc. This step was called cleaning step. The next step in this tier was the transliterates step. Since we were reading the text from one Azeri text, some sources encoded the text in Cyrillic letters. It may not seem obvious at first, because the letter "a" in both cases were identical but their character codes were different. Hence, the space they capture was different. The string was separated into sentences, and sentences were separated into lists of the words. Those lists of words were input parameters to the model. Speaking of the model, the second tier was Markov Model. This was where the actual processing and predicting happens (Fig. 2). Markov Model preserves 3 databases to capture relations. While looping through the words of the sentences, it captures the relations between maximum 3 words sequences. 2 words and the next words were saved in a dictionary. Number of times those 2 words sequences occurred, it increased the probability of predicting the next word if these words were typed. Next, apart from those 3 words sequences, 2 words sequences were also captured. Those were one word followed by the next word type of sequences and probability of those were computed as well. We also saved the initial words of each sentence to have an initial prediction rather than waiting for the user to initiate the process. After processing's done, the model is serialized and saved to reduce the time for computation which is already done just like the way all modern APIs follow. Markov Model tier exposes only 2 methods, one to get the trained instance, the next one is the method to predict next words. The final tier was the UI. This was where the user interacts with the system. It utilizes the next words method of the Markov Model (Fig. 3).

III. Collection of The Database

The software of this research was included a small Azeri vocabulary database where the amount of it was about 4 MB. it was constructed by a predictor parser. The database was concluded by the most general Azeri language words to introduce HMMs based generated word pairs. In the Step of Data Collection for HMM Model, Web Scraper was developed to obtain essential training data for the model, using Python and Selenium Web Browser Automation tool. The file "MAIN.PY" of the software, was responsible for establishing a connection with a website that is needed to scrape and perform required actions which it was concluded:

- **webdriver:** is an object of Selenium tool which utilizes incognito mode in Google Chrome. To launch Chrome Browser, executable path of webdriver should be passed as a parameter to webdriver.Chrome() function.

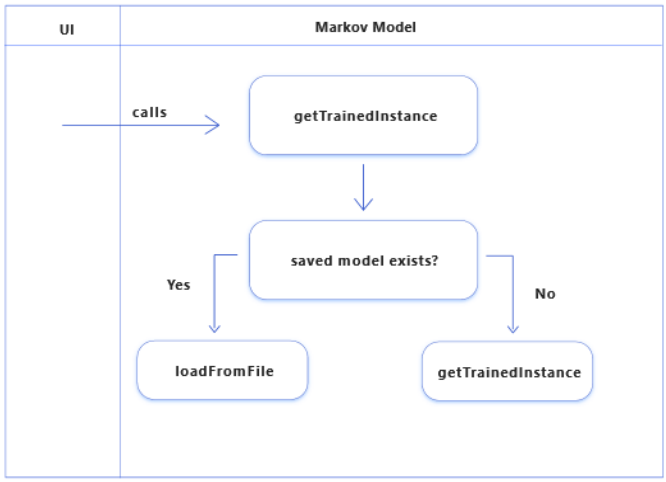

Fig. 1. The Preprocessing tier

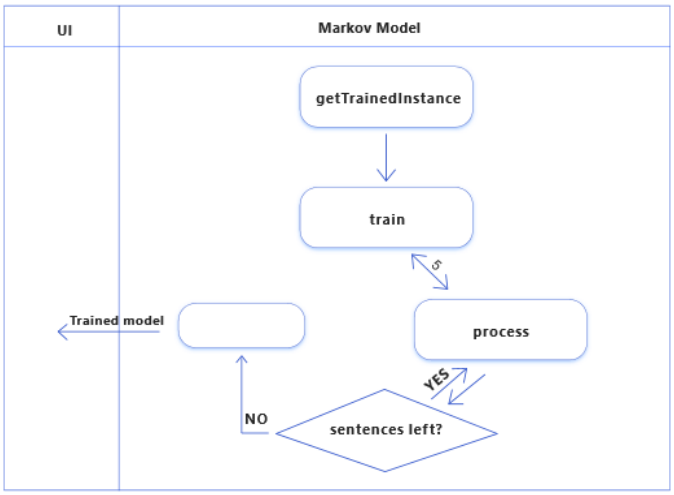

Fig. 2. The Predicting tier

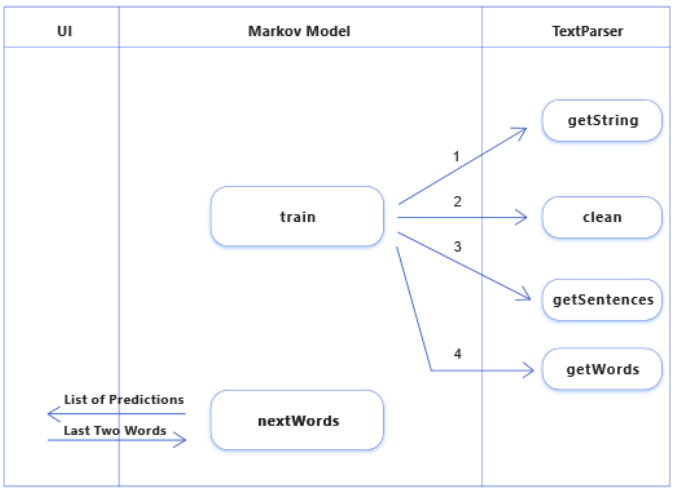

Fig. 3. The UI tier

- **browser**: is a variable that holds value returned from webdriver.Chrome().
- **browser.get("URL")**: method is used to get desired webpage. Subsequently the code in Fig. 4 is used to

determine whether a website is loaded successfully, by checking on XPATH value of an HTML attribute, within predefined time interval.

- **browser.find_element_by_xpath("XPATH").click( )**: is used to automate a click action on an HTML element to exploit the content it holds. Then it is assigned to a new, post-variable.
- **post.text**: Fetches text content of post object.

```
timeout = 10
try:
    WebDriverWait(browser, timeout).until(
        EC.visibility_of_element_located((By.XPATH, """XPATH OF A PARTICULAR HTML
ELEMENT HERE""")))
except TimeoutException:
    print("Timed out waiting for page to load")
    browser.quit()
```

Fig. 4. **browser.get("URL")** method

IV. The Software

The software was included a small Azeri vocabulary database where the amount of it was about 4 MB, the various Python libraries, a HMM model and a Web based interface (GUI). This software was included these 5 files:

- Main.java
- Main_VC.java
- Main_Model.java
- Main_View.css
- EditorView.fxml

MAIN.JAVA file is a driver class for the project. It has 2 functions as:

- main (String[] args) - Launches the program.
- start (Stage stage) - Loads the FXML file, sets the main view controller, Sets new scene (loads the content of the FXML to this scene), sets window's resizability to false, and shows the main view stage.

MAIN_VC.JAVA file is the view controller of the main scene. It connects EditorView.fxml to the Main_Model.java . This class implements the Initializable class. There are 3 variables:

- model - It is a private variable, its data type is Main_Model, and basically connects this class to Main_Model class.
- listView - It is a private variable, its data type is ListView<String>, and it is connected to the EditorView.fxml via @FXML.
- textArea - It is a private variable, its data type is TextArea, and it is connected to the EditorView.fxml via @FXML.

This class has 3 functions:

- Main_VC(Main_Model model) - It's a public function, it is a constructor, and sets a value the model.
- onCellClicked() - It's a public function, its output is void, it is connected to the EditorView.fxml via @FXML, it is called when a cell of the listView is clicked via mouse, it gets text from the selected cell of the listView. If it is a ". ", the last character (which is " ") is erased from the textArea. Sets the text of the textArea to the selected word from the listView. Adds an " " to the end. Then updates the listView.
- initialize(URL url, ResourceBundle rb) - It's a public function, its output is void, is called when the scene is loaded, and adds an event listener to the textArea, so when the space key is pressed the listView is updated.

MAIN_MODEL.JAVA file class has one variable:

- hmm - It is a private variable, its data type is MarkovModel, and its initial value is a new instance of MarkovModel class.

This class has 3 functions:

- loadData(TextArea textArea, ListView listView) - It's a public function, its output is void, it gets the string value of the textArea, makes it lowercase, gets an array of 2 strings using getLastTwo function, gets an arrayList of predicted words using hmm's nextWord function, clears the listView, and gets top five predictions using getFive function and adds them to the listView.
- getLastTwo(String input) - It's a public function, its output is an array of strings, and returns an array of the last to words of a given string.
- getFive(List<String> list) - It's a public function, Its output is a list of strings, and it returns a list of first 5 elements of the given list.

MAINVIEW.CSS file is a stylesheet for the FXML file. It helps to modify the attributes of some elements that are not modifiable within "Scene Builder". It was used 3 main colors (lightest to darkest): #E6F0D9, #7A817B, and #151721. and the font is "Exo".

EDITORVIEW.FXML is a view file for the main scene that is created in Gluon's "Scene Builder". Our scene has a TextArea and a ListView. Also, on top there is a HBox with a Label inside.

V. Some Experimental Results

A GUI was implemented to test the algorithm. After writing an Azeri word in the "Text Area" part of the GUI, they will be predicted 5 the highest probability next words and will showed in the "List of The Top Predictions (max 5)" part of the GUI (Fig. 5). Using the GUI, the prediction accuracy was evaluated. The Model was trained by 90% of the database (Fig. 6). Then, predicting the next 5 words on the train data resulted 100% accuracy and predicting the next 5 word on the test data resulted 54% accuracy (Fig. 7).

VI. Conclusions

This paper tackled the problem of HMMs based next word prediction for Azeri (Azerbaijani) language, based on NLP principles using Python programming language. The software was included a small Azeri vocabulary database where the amount of it was about 4 MB, the various Python libraries, a HMM model and a Web based interface (GUI). In this research, the database was constructed by a predictor parser which it was implemented for the first time for Azeri (Azerbaijani) language. The database was concluded by the most general Azeri language words to introduce HMMs based generated word pairs. The Model was trained by 90% of the database, hence, predicting the next 5 words on the test data resulted 54% accuracy.

Fig. 5. The implemented GUI to test the algorithm

Fig. 6. Training of the model by 90% of the database

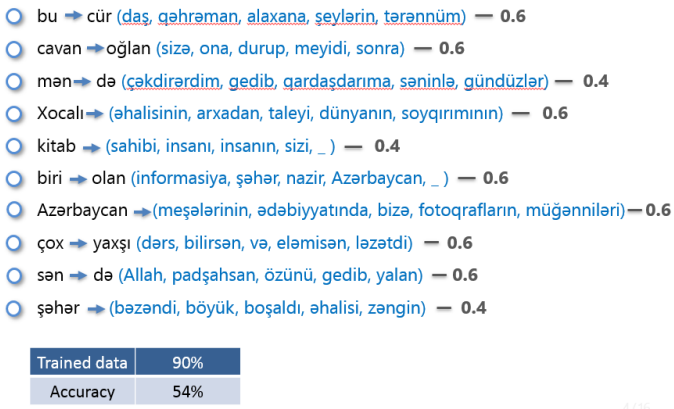

Fig. 7. Predicting the next 5 words on the test data

REFERENCES

[1] A.A. Kibrik, E.R. Tenishev, E.A. Poceluevskij and I.V. Kormushin, “Languages of the world: Turkic languages,” Jazyki mira: Tjurkskie jazyki, Moscow: Indrik, 1997, pp.542.

[2] A. Abbasov, R. Fatullayev, A. Fatullayev, “HMM-Based Large Vocabulary Continuous Speech Recognition System For Azerbaijani,” The Third International Conference on Problems of Cybernetics and Informatics, Baku, Azerbaijan, September 6-8, 2010, pp.23-26.

[3] K.R. Aida-zade, C. Ardil, S.S. Rustamov, “Investigation of Combined use of MFCC and LPC Features in Speech Recognition Systems,” IJSP: International Journal of Signal Processing, 2006, V. 3, pp.105-111.

[4] R. Fatullayev, A. Abbasov, A. Fatullayev, “Dilmanc is the 1st MT system for Azerbaijani,” In: Proc. of SLTC-08, Stockholm, Sweden, 2008, pp.63-64.

[5] A.M. Sharifova, V.A. Dadalov, I.E. Ibrahimov, “Text Normalization System for Azerbaijan TTS,” In: Proc. of International Symposium on INnovations in Intelligent SysTems and Applications (INISTA 2009), Trabozan, Turkey, 2009, pp.71-74.

[6] K.R. Aida-zade, S.G. Talibov, “Analysis of the effectiveness of the methods of recognition of authorship of texts in the Azerbaijani language,” In: Proc. of The 5th International Conference on Control and Optimization with Industrial Applications, Baku, Azerbaijan, 27-29 August, 2015, pp.183.

[7] P.P. Barman, A. Boruah, " A RNN based Approach for next word prediction in Assamese Phonetic Transcription," 8th International Conference on Advances in Computing and Communication (ICACC), Procedia Computer Science, 2018, 143, pp.117–123.

[8] F.A. Gers, J. Schmidhuber, F. Cummins, “ Learning to forget: Continual prediction with lstm,” 1999.

[9] J. Luis Garcia Rosa, "Next word prediction in a connectionist distributed representation system," IEEE International Conference on Systems, Man and Cybernetics, Yasmine Hammamet, Tunisia, 2002, pp. 6 pp. vol.3-.

[10] T. Mikolov, M. Karafiat , L. Burget, J. Cernocky, S. Khudanpur, “Recurrent neural network based language model,” in: Eleventh Annual Conference of the International Speech Communication Association, 2010.

[11] S. Sukhbaatar, J. Weston, R. Fergus, etal.,”End-to-end memory networks,” in: Advances in neural information processing systems, 2015, pp.2440–2448.

[12] M. Panzner, P. Cimiano, “Comparing Hidden Markov Models and Long Short Term Memory Neural Networks for Learning Action Representations,” In: Pardalos P., Conca P., Giuffrida G., Nicosia G. (eds) Machine Learning, Optimization, and Big Data. MOD 2016. Lecture Notes in Computer Science, vol 10122. Springer, Cham.

[13] L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proc. of IEEE, 1989, 77(2), pp.257-286.