Vol. 2, No. 2 | July – December 2018

Effective Word Prediction in Urdu Language Using Stochastic Model

Muhammad Hassan* Muhammad Saeed* Ali Nawaz* Kamran Ahsan†
Sehar Jabeen* Farhan Ahmed Siddiqui* Khawar Islam†

Abstract

Word prediction and word suggestion is an important tools for writing contents in any language, in which the right word is predicted in a current context. Writing contents on English keyboards to produce other language contents is too hard and time consuming for everyone and requires more practice. To increase the typing speed especially in mobile and smart phones or for creating contents on social networks derived the need of this tool in every language. This paper presents a state of the art research for word prediction in Urdu Language (UL) based on stochastic model. Hidden Markov Model was implemented to predict the next state, while Unigram Model was also used to suggest the current state and the next hidden state, N-Gram Model was followed keeping N=2. The tool is developed to implement this model for Urdu Language (UL) and tested by regular and new URDU content writers to check their improvements in their typing speeds.

Keywords: *Word Prediction, Natural language processing, stochastic model, unigram, interpolation, markov model*

1. Introduction

Word Prediction (WP) is a significant task of Natural Language Processing (NLP) and probability theory. It is intended to predict the right word in a given context. Word prediction can be utilized in numerous applications. For instance, prescient content section frameworks, word consummation utilities, and composing helps [1] and [2]. Many prediction methods are available in IT industry. The well famous systems are T9(), eZiText(), iTAP(), and these systems are adopted by large smartphones companies. This paper presents a model, a word prediction application for Urdu language. After typing one word, the model gives a suggestion wordlist to a user which user wants to write and think about the next word. Since word prediction has always become important task for users to minimize their keystrokes while typing and provide suggestions to use different words according to the context. A typical way to handle and deal with this problem is to prepare and apply stochastic approach based on Hidden Markov Model (HMM) and Unigram Model (UM).

As there isn't any previous work done for Urdu Language (UL), the state of the art work presents a successful implementation for word prediction utilizing probabilistic model and selected techniques. N=2 is selected for the current state of suggested words. Probabilistic techniques have been applied on Urdu Language (UL) to obtain computable linguistic artifacts. Most of Pakistan's 190 million population can speak URDU but while typing they are not fluent and feel difficulty [3]. The study covers the following aspect which will target the technical community and make their work easy to make new applications for users on every platform by providing the features of this study in several ways like Desktop Applications, Mobile Applications, Cross Platform Applications, Web and Online Applications, etc.

Today all existing smartphone or portable devices have predictive keyboard [4]. This keyboard helps in typing by predicting the next word and completing word suggestion which aids in faster typing and time saving [5]. Although, sometimes results are sidesplitting as the keyboard is not perfect [6],to train soft keyword is not very difficult. [4] used word-predicting technology to suggest a text entry in soft keyboard, but this feature can generate incorrect suggestions – especially when typing is being done in another language or slang. There are also many little suggestions and prediction applications available, but those are for users who can run their applications on fixed platforms only.

*Department of Computer Science, University of Karachi, Pakistan
†Department of Computer Science, Federal Urdu University of Science & Technology, Karachi

2. Literature Review

Various techniques and frameworks have been proposed for word prediction in the previous couple of decades. These techniques could be described through stochastic model and Markov Model (MM) predicting the future state with the help of unigram techniques. In [4] presented Urdu corpus character frequency analysis. The Monte Carlo Simulation performed with simulated annealing to optimize the keyboard layout. Moreover, the keyboard layout was augmented for speeding up the text entry from Urdu corpus lists to predict text. For justification purpose, the keyboard performance analysis was done. Carlo et al [7] presented a FastType prediction system based on statistical and lexical methods. [8], proposed an effective work derived from surface features for the accurate prediction of word. [9] presented a new approach called relational feature for predicting word. In which each word was treated as a word and then predicted in its context. [10] implemented stochastic model for the word prediction in Bangla language. They used a large corpus, reduced the chance of misspelling and implemented choices of the right word according to current context. [11] proposed a new technique for automatic word prediction based on content mining specially designed for the big data systems. Kenneth et al [12] introduced an extension for smartphone keyboard. When user tapped on keyboard, different phrase suggestions appeared on user keyboard. [13] presented string matching word prediction by implementing Morris Pratt algorithm. This technique was evaluated using text classification and several more techniques. [14] worked on crowdfunding websites (Entrepreneurs & Artists) and predicted different phrases through the study of different projects, analyzed predictive power of phrases and constructed dataset for phrase prediction. In [15], proposed a probabilistic driven model to predict word as well as phrase. They introduced FussyTree structure to address both problems.

2.1 Slow Typing Speed

Many significant challenges and issues are being faced in Urdu Language (UL) in Urdu typing. A Survey conducted by (10fastfingers) URDU typing speed is 50% less than the English typing speed that was just because of double characters per key on the keyboard which caused user to avoid the language for typing purpose and instead use another. Typing competition was conducted by (10fastfingers) for Urdu and English typing speed, the Urdu typist was around 70+WPM, and whereas in English typist's speed was about 160+ WPM.

2.2 Collecting Urdu Words

In this section, we discussed the problems faced in collecting Urdu words. We collected Urdu words from different sources [16], [17], and [3]. From a corpus of source-target sentence, we wrote many programs that perform different phases in order to extract only the Urdu syntax words. Firstly, for good predictive suggestion, collecting the words was a major challenge [18] and [4]. We designed little software for crawling, extracting, cleaning, counting and more to crawl the websites and then extracted only the Urdu syntax words, then cleaned all the other languages other than Urdu and then counted their occurrence on repeating to find out their frequency. We were able to find out 1.25Lac+ unique Urdu words that was a big data till now in its own category. Now for the other category of double words collection, we again crawled the internet and browsed to be able to find out about 0.5Lac+ double words with their own most used general collection. Two-big data helped us out to work further and gave the best performance that it could.

3. Procedure

3.1 Selecting Algorithm

Selection of algorithm was a challenging part, we can get our start by analyzing and writing many algorithms. We ended up with an optimized version of algorithm that would fulfill our need of output with the best utilization and artificial intelligence with work behind the screen and to make the intelligence better for front-end program to get experience that is more professional with the application.

3.2 Urdu Suggestion and Prediction

The writing breakdown of Urdu Language text can be eased by an "Intelligent" word predicting feature of word processing. Due to 50+ alphabet, and 50% of them being used with SHIFT keys, the range was controlled by reducing the keystrokes necessary for word typing. The word prediction monitored the user input as s/he types letter by letter and suggested the appropriate word list with beginning letter or contained the sequence of input letters [19] and [20]. The list was updated by input of each letter. The desired words were chosen from the suggested word list, which was being inserted in the cursor position in the ongoing sentence or text just by a single keystroke or selection. Usually, the list of words was numbered and could be entered by typing the respective number.

- Match typed word from the prefix of dictionary words
- Predict dictionary typed words that user just typed before
- Suggest wrong word that user type, but not present in the dictionary
- Get the frequency of the character to match them first
- Suggest the closed typed word again first

Table 1: Training Algorithm for Predicting Urdu Language Sentences

```
HashMap ← SingleWordDictionary
HashMap ← DoubleWordDictionary
List ← UserSingleWordDictionary
List ← UserDoubleWordDictionary
Initialize the data
If (User Type)
    Var ← All Text
Position ← Type Position
Var ← Last Char Before Caret
Dictionary (Last Char Is Alphabet)
    Var ← Get That Whole Word till Last Space
    Var ← Garb All the Text
    Array ← Extract All the Grams
    Dic ← Update The Runtime Dic
    Sort ← Sort as Per Frequency
    Find ← Match the Starting Word From Runtime Dic
    Find ← Match the Starting Word From Dic
    Extract ← Filter the Words from the DICs
    Final ← Make a Final List
    Show ← Return to the User
Else If (Last Char Is Not Alphabet)
    Do the Same About but With Double Word Dic
OnClosing, Save The Words.
Step 1: Initialize single & double Urdu words at class level.
Step 2: Check the text changed event in textbox.
Step 3: Get the caret position.
Step 4: Pick up the last char before caret.
Step 5: Check
IF last char is an alphabet, pick up that's alphabet.
ELSE IF last char is anything else then first pick up last full word.
Step 6: Get all the text from the text box.
Step 7: Split it with special char.
Step 8: Get all words in an array.
Step 9: if the last char is holding.
Step 10: Match that char with the starting from the runtime dictionary.
Step 11: Get some words that are in runtime dictionary not in main dictionary.
Step 12: Get some words that are in runtime dictionary and also in main dictionary
Step 13: Get some words that are in main dictionary
Step 14: Now get the position of the caret.
Step 15: Show the collected matched word list there.
Step 16: Now if the last word is holding.
Step 17: Match that word with the starting from the runtime dictionary.
Step 18: Get some words that are in runtime dictionary not in main dictionary.
Step 19: Get some words that are in runtime dictionary and also in main dictionary.
Step 20: Get some words that are in main dictionary.
Step 21: Now get the position of the caret.
Step 22: show the collected matched word list
```

4. Experimental Steps

In this section, the features and techniques of the methodology are discussed. The methodology is clear and straight as the intelligent words were built with the implementation of artificial intelligence, through which the user typed character and a list of suggestions to complete the word is available for the user. When the user types a completed word, then the next most probability holder word would be given to be fitted there so it could save time more than before with 99% perfect correction. The study ended up with an optimized version of the algorithm that would fulfill the need of output with the best utilization and artificial intelligence by working behind the screen as well to make the intelligence even better for user of front-end program to get experience that is more professional with the application.

4.1 Software Flow Diagram

The flow of working starts with the typing of the first character and moves forward with the subsequent input. Different states and different variations were considered and described with parallel conditions, activities and system functions to achieve the goal. Figure 1 shows the outlook view of algorithm.

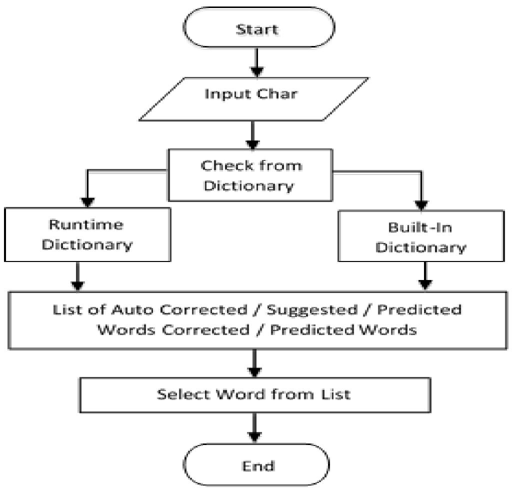
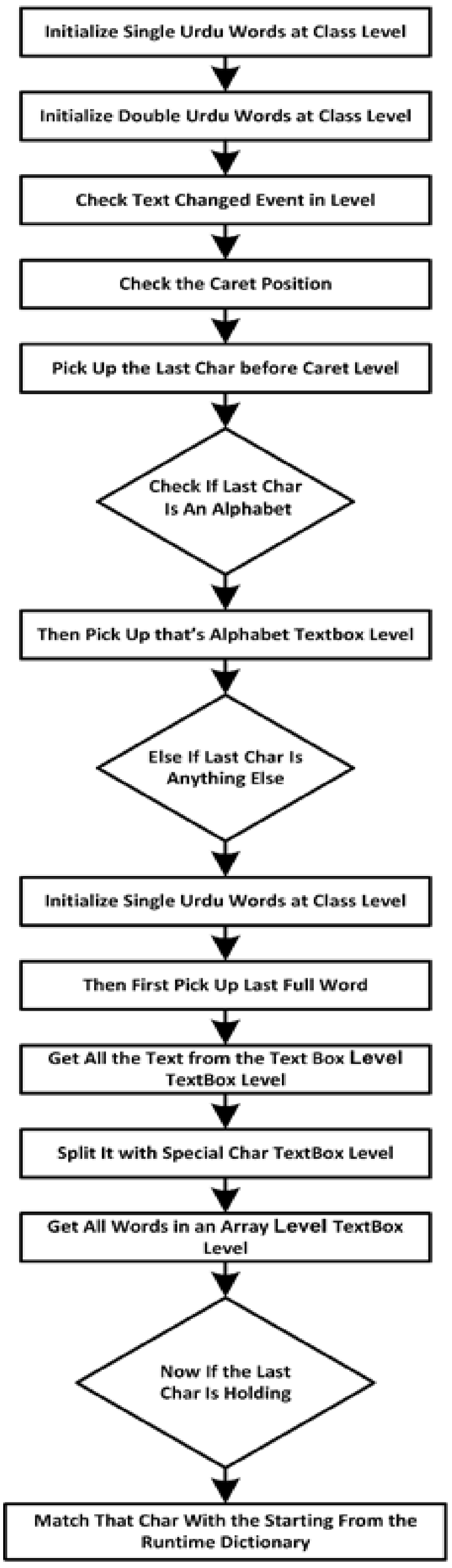


Figure 1: Representational Outlook of Idea

4.2 Diagrammatical View of Algorithm

The Diagrammatical view of the optimized version of the algorithm that would fulfill the need of output with the best consumption and artificial intelligence, with working behind the screen, which is mentioned above, is presented in Figure 2.



Initialize Single Urdu Words at Class Level
Initialize Double Urdu Words at Class Level
Check Text Changed Event in Level
Check the Caret Position
Pick Up the Last Char before Caret Level
Check If Last Char Is An Alphabet
Then Pick Up that's Alphabet Textbox Level
Else If Last Char Is Anything Else
Initialize Single Urdu Words at Class Level
Then First Pick Up Last Full Word
Get All the Text from the Text Box Level TextBox Level
Split It with Special Char TextBox Level
Get All Words in an Array Level TextBox Level
Now If the Last Char Is Holding
Match That Char With the Starting From the Runtime Dictionary

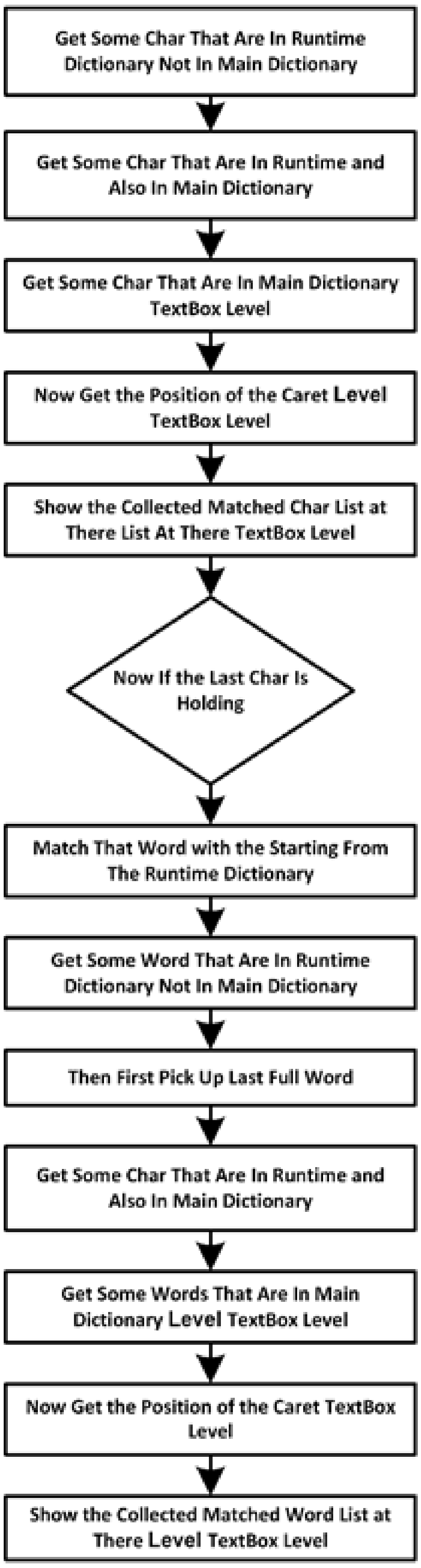

Get Some Char That Are In Runtime Dictionary Not In Main Dictionary
Get Some Char That Are In Runtime and Also In Main Dictionary
Get Some Char That Are In Main Dictionary TextBox Level
Now Get the Position of the Caret Level TextBox Level
Show the Collected Matched Char List at There List At There TextBox Level
Now If the Last Char Is Holding
Match That Word with the Starting From The Runtime Dictionary
Get Some Word That Are In Runtime Dictionary Not In Main Dictionary
Then First Pick Up Last Full Word
Get Some Char That Are In Runtime and Also In Main Dictionary
Get Some Words That Are In Main Dictionary Level TextBox Level
Now Get the Position of the Caret TextBox Level
Show the Collected Matched Word List at There Level TextBox Level

Figure 2: Flow Chart

4.3 Data Structure of UL

We used two data structures to maintain Urdu dictionary. We implemented word suggestion and prediction using unigram and bigram models. The list of the data structures was maintained and used for different purposes. It was needed to maintain a list of all unique words with associated index values in the Urdu corpus. HashMap was suitable, because I would keep the RAM compressed and clean to avoid leaving little empty spaces behind in RAM because if it is required to search any last character list word then it is not needed to crawl all other characters before as it was done before so it became better and optimized also in view of computer load and performance.

4.4 Current Dictionary Prediction and Suggestion

It is now made possible to show words from URDU dictionary starting from the first letter that are typed in ascending order and showed the next word as per the last typed word from the most common dictionary.

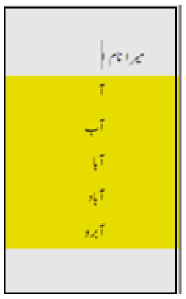


Figure 3: Current Dictionary Predict & Suggest

4.5 Current Runtime Prediction and Suggestion

It is now made possible to show words from URDU dictionary starting from the first letter that were typed in ascending order but will show the user's typed dictionary word on the top of the list and will also show the next word as per the last word typed by the user from the current runtime study from the dictionary.

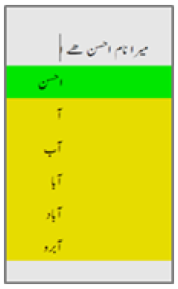


Figure 4: Current Runtime Predict & Suggest

4.6 Current Typed Text Prediction and Suggestion

It is made possible to show words from URDU dictionary starting with the first letter that were typed in ascending order and would show the user the typed dictionary word on top of the list and also show the wrong typed words in the list also and would will also show the next word as per the last typed word from the current runtime study.

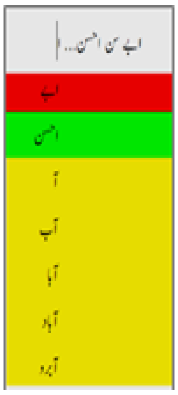


Figure 5: Current Typed Text Predict & Suggest

4.7 Next Dictionary Prediction and Suggestion

Through the study it is now made possible to show words from URDU dictionary starting with the first letter that were typed in ascending order and it would also show the next word as per the last typed word from the most common dictionary.

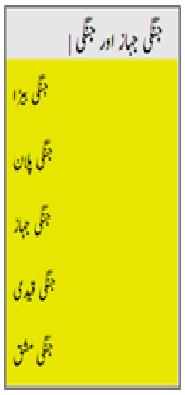


Figure 6: Next Dictionary Predict & Suggest

4.8 Next Runtime Prediction and Suggestion

The study made it achievable to show words from URDU dictionary starting with the first letter that were typed in ascending order but would show the typed dictionary word on the top of the list and also show the next word as per the last typed word from the current runtime study from the dictionary.

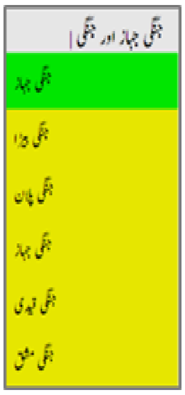


Figure 7: Next Runtime Predict & Suggest

4.9 Next Typed Text Prediction and Suggestion

The study showed words from URDU dictionary starting with the first letter that was typed in ascending order and would show the typed dictionary word on the top of the list and also show the wrong typed words in the list as well and it would also show the next word as per the last typed word from the current runtime study

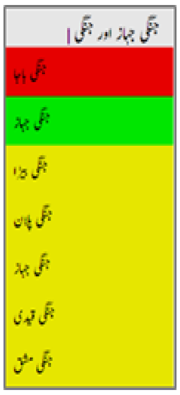


Figure 8: Next Typed Text Predict & Suggest

4.10 N-gram Model

In statistical based techniques, by performing n-gram analysis, the sentence probability was to be measured. Here if the text was split in double words, it ended up in (N/2) words lists of the total (N) text but this would not be intelligent as it should be so that if there is a text "I am a student", then here as per (N/2) logic, it would get "I am", "a student" that means that when it was typed "I" then it would have got "am" as suggestion to next word and same as "a" word would get "student" as a suggestion but what about when it was typed "am" then as per rule, it should have got "a" as suggestion but (N/2) logic would not help in this phase so it was not recommended. So for taking hold on this problem, (N-1) logic was used to cover this problem so here the text was split in double words then it ended up in (N/2) words lists of the total (N) text as it should as if there is a text “I am a student”, then here as per (N-1) logic, it would get “I am”, “am a”, ”a student” that means this would be intelligent as it should, as if there was a text “I am a student”, then here as per (N-1) logic, it would get “I am”, ”a student” that mean when it was typed “I” then it would get “am” as a suggestion for the next word and same as “a” word will get “student” suggestion and when it was typed “am” then it would also get “a” as a suggestion as per the new rule.

4.11 Hidden Markov Model

The classification problems could be solved by utilizing the Hidden Markov Model (HMM); one of the statistical based model presented by [21] and [22]. It could be represented as the set of transition probabilities connecting the interconnected set of states. So here when there are BiGrams, the previous states from where the user has passed to make a map of the states out of a full word was watched, so later when keeping the frequencies of the transaction from the states, the next typed words is redirected as per the transaction probability secured earlier.

4.12 Searching from Main Dictionary

A list was first created to keep the default dictionary data that was good over array because it would keep the RAM compressed and clean to avoid leaving little empty spaces behind in the RAM but this data list became long that was also good but the Linear Search would go to worst case to O(n) that would take time to search at every click of the mouse. To avoid this, this concept was avoided so the logic that was the same as a HASH Table or HASH Map etc. was created. One list of dictionaries was simply divided in 40 dictionaries and assigned each dictionary to an alphabet/character of URDU that leads us to make our worst-case O(n)/40 that meant that it got 40% faster than before. Here just the desired word starting character needed to be checked and redirected the search to only that character dictionary list and whatever it would find there, it would be much faster than before. If it was required to search any last character list word, it was not required to crawl all other characters before as it was done, So here it got better and optimized also in view of computer load and performance.

4.13 Searching from Run-Time Dictionary

It is now made achievable to show words from URDU dictionary starting from the first letter that was typed in ascending order but would show the typed dictionary word on top of the list and it would also show the next word as per the last typed word from the current runtime study from the dictionary.

5. Overview of Modules

The study has designed a GUI application which contains a text fields where user would type word and application would give suggestion so that the user would be able to get access to the suggestion lists in an easy way and it also initialized all the single word dictionary and double word dictionary at the time of object creating. It was required to show the suggestion so after finding out the position, the user just went one char back to find out what user typed and then user chooses from the possibilities that either they typed an alphabet char or a non-alphabet char that would help them by taking the right decision.

5.1 Updating Run-Time Dictionaries

If there would be a non-alphabet character, the user would run the background process that would update the run time dictionary with user for the currently typed words. It was also required to make the double word dictionary at the run time but that would be a different task as normal. It was attempted to create a full probability approach to cover all the maximum probability as shared above under “Using (N-1) Logic for More Artificial Intelligence”.

5.2 Half or Full Typed Words

After getting double word runtime dictionary, programmer would be able to get the user data in their program that would help them later to suggest the better and perfect. Till now there was a fully typed word and a half-typed word, it was again split in two different ways to do as per the requirements.

6. Results and Discussion

This paper presented a model which was successfully implemented using C# and tested as a standalone model. This model was based on Urdu language which contains two dictionaries called pre-build and runtime word list. Upon user typing, the model predicted new words with its current context and store into the pre-build dictionary and display the word list as a suggestion. The runtime dictionary shows those words which are nearest to the user typing and predicted from the main dictionary. User easily accessible all the words available in runtime wordlist. This model also achieved some additional feature where user easily manage each word by pressing key up and down arrow in the word list using keyboard. The user easily press enter key to select any word given in word list. The design of the model is simple and easily used by any user, programmer and, etc. We set up different parameters (words) to validate our model as a standalone application. The predicted words were accurate and nearest to the user context. This model adds new contribution in Urdu language and software industry.

7. Application Overview

An applications was built as mentioned above. Some working screenshots are taken from it to explain it more clearly.

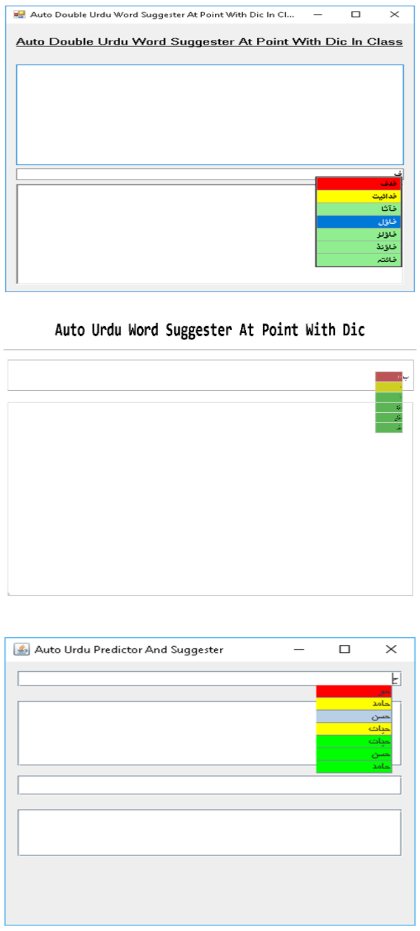

Auto Double Urdu Word Suggester At Point With Dic In Cl...
Auto Double Urdu Word Suggester At Point With Dic In Class
Auto Urdu Word Suggester At Point With Dic
Auto Urdu Predictor And Suggester

Figure 9: Application View

8. Conclusion

A pioneering research has been presented for Auto Suggestor and Predictor of word (Get Your Word Before Your Typing to Increase The Typing Speed).(N-1) logic has been used for more Artificial Intelligence, different algorithms, and searching approaches to increase the typing speed by providing better auto suggestions and predictions. Although the research and experiments did not contain a huge amount of data, the outcomes met state-of –the- art work. It has been noticed that words dictionary, quality and quantity had significant impact on predicting the list of suggestion and its precision.

References

[1] Malik, A.A., and Habib, A.: 'Urdu to English machine translation using bilingual evaluation understudy', International Journal of Computer Applications, 2013, 82, (7)

[2] Manchanda, B., Athavale, V.A., and kumar Sharma, S.: 'Various Techniques Used For Grammar Checking', International Journal of Computer Applications & Information Technology, 2016, 9, (1), pp. 177

[3] Asif, T., Ali, A., and Malik, K.: 'Developing a POS tagged Resource in Urdu', in Editor (Ed.): 'Book Developing a POS tagged Resource in Urdu' (Punjab University College of Information Technology, University of the Punjab, 2012, edn.), pp.

[4] Khan, M.A., Khan, M.A., and Ali, M.N.: 'Design of Urdu Virtual Keyboard', in Editor (Ed.): 'Book Design of Urdu Virtual Keyboard' (2009, edn.), pp. 126-130

[5] Kabir, H.: 'Two-pass parsing implementation for an Urdu Grammar Checker', in Editor (Ed.): 'Book Two-pass parsing implementation for an Urdu Grammar Checker' (IEEE, 2002, edn.), pp. 51-51

[6] Malik, A.A., and Habib, A.: 'Qualitative Analysis of Contemporary Urdu Machine Translation Systems', in Editor (Ed.): 'Book Qualitative Analysis of Contemporary Urdu Machine Translation Systems' (Citeseer, 2013, edn.), pp. 27-36

[7] Aliprandi, C., Carmignani, N., and Mancarella, P.: 'An inflected-sensitive letter and word prediction system', International Journal of Computing & Information Sciences, 2007, 5, (2), pp. 79-85

[8] Li, J.J., and Nenkova, A.: 'Fast and accurate prediction of sentence specificity', in Editor (Ed.): 'Book Fast and accurate prediction of sentence specificity' (2015, edn.), pp.

[9] Even-Zohar, Y., and Roth, D.: 'A classification approach to word prediction', in Editor (Ed.): 'Book A classification approach to word prediction' (Association for Computational Linguistics, 2000, edn.), pp. 124-131

[10] Haque, M., Habib, M., and Rahman, M.: 'Automated word prediction in bangla language using stochastic language models', arXiv preprint arXiv:1602.07803, 2016

[11] Horecki, K., and Mazurkiewicz, J.: 'Natural language processing methods used for automatic prediction mechanism of related phenomenon', in Editor (Ed.): 'Book Natural language processing methods used for automatic prediction mechanism of related phenomenon' (Springer, 2015, edn.), pp. 13-24

[12] Arnold, K.C., Gajos, K.Z., and Kalai, A.T.: 'On suggesting phrases vs. predicting words for mobile

text composition', in Editor (Ed.): 'Book On suggesting phrases vs. predicting words for mobile text composition' (ACM, 2016, edn.), pp. 603-608

[13] Knoll, B.: 'Text Prediction and Classification Using String Matching', Journal of Machine Learning Research, 2002, 2, pp. 419-444

[14] Mitra, T., and Gilbert, E.: 'The language that gets people to give: Phrases that predict success on kickstarter', in Editor (Ed.): 'Book The language that gets people to give: Phrases that predict success on kickstarter' (ACM, 2014, edn.), pp. 49-61

[15] Nandi, A., and Jagadish, H.: 'Effective phrase prediction', in Editor (Ed.): 'Book Effective phrase prediction' (VLDB Endowment, 2007, edn.), pp. 219-230

[16] Hardie, A.: 'Developing a tagset for automated part-of-speech tagging in Urdu', in Editor (Ed.): 'Book Developing a tagset for automated part-of-speech tagging in Urdu' (2003, edn.), pp.

[17] Abbas, Q.: 'Morphologically rich Urdu grammar parsing using Earley algorithm', Natural Language Engineering, 2016, 22, (5), pp. 775-810

[18] Jawaid, B., and Zeman, D.: 'Word-order issues in english-to-urdu statistical machine translation', The Prague Bulletin of Mathematical Linguistics, 2011, 95, pp. 87-106

[19] Ijaz, M., and Hussain, S.: 'Corpus based Urdu lexicon development', in Editor (Ed.): 'Book Corpus based Urdu lexicon development' (2007, edn.), pp.

[20] Iqbal, S., Anwar, M.W., Bajwa, U.I., and Rehman, Z.: 'Urdu spell checking: Reverse edit distance approach', in Editor (Ed.): 'Book Urdu spell checking: Reverse edit distance approach' (2013, edn.), pp. 58-65

[21] Anwar, W., Wang, X., Li, L., and Wand, X.: 'Hidden markov model based part of speech tagger for urdu', Information Technology Journal, 2007, 6, (8), pp. 1190-1198

[22] Gupta, N., and Mathur, P.: 'Spell checking techniques in NLP: a survey', International Journal of Advanced Research in Computer Science and Software Engineering, 2012, 2, (12)