IJCTA, 9(22), 2016, pp. 115-122

Word Prediction using Collaborative Filtering Algorithm

Soumalya Ghosh* Hukam Singh Rana Ravi Tomar*****

Abstract : Word prediction refers to predict the most probable next word which is intended to be typed afterward. So, it enables the user to compose a word without completely typing all the characters of it. Hence it is facilitated to save the number of required keystrokes. However, development of an efficient word prediction system is very much obstructed due to the data sparsity in training data. The paper will be going to address the negative effect of the data sparsity on word prediction. The performance of traditional statistical word prediction techniques like unigram bigram, n-gram etc., are bounded due the to data sparseness. That can be resolved by any advance data cleaning techniques like smoothing, filtering etc. Here in this paper we use collaborative filtering technique to address the issue of data sparsity and enhance word prediction quality. The proposed approach is established through critically evaluated with the different considered metrics like percentage of key strokes required to predict the intended word, percentage of key strokes save by using the prediction engine, and percentage of accuracy of our word prediction model. The empirical evaluation has been proven that the proposed approach is superior than the traditional statistical word prediction technique.

Keywords : Word Prediction, Data Sparseness, Smoothing, Collaborative filtering, Pearson correlation coefficient.

1. INTRODUCTION

Word prediction mechanism is one of the foremost text entry rate enhancement strategy which is implemented through predicting most the probable next word which a user is going to be typed. During the typing, the prediction system monitors each input or typed character one by one and a list of most probable words based on the previously typed sequence is created. The list of the probable words is updated whenever a character is added or removed from the typed sequence. When the list shows the required intended word, it is selected and inserted into the text which is under composition. Some of the advantages of any generic word prediction system are listed in the following section.

1. **Improvement of text entry rate :** The word prediction system eliminates the need to type a complete textual chunk and thus improving or enhancing the text entry rate.
2. **Reducing effort :** The word prediction system saves the keystrokes required to compose a complete text, as it predicts the whole word after typing a few characters and thus the effort requires for typing is reduced.
3. **Spell checking :** The word prediction system is identified the intended word after typing first few characters of that word. It can predict the correct one, even if there may also be some misspelled characters in the typed sequence. So, it helps to write the correct spelling of the entire word.

* Center for Information Technology University of Petroleum and Energy Studies Dehradun, India sghosh@ddn.upes.ac.in
** Center for Information Technology University of Petroleum and Energy Studies Dehradun, India hsran@ddn.upes.ac.in
*** Center for Information Technology University of Petroleum and Energy Studies Dehradun, India rtomar@ddn.upes.ac.in

The advantage of the word prediction system is not only restricted on the previously pointed list. It is also advantageous for the persons with language impairments and physical disabilities. This can be done by improving the quality of the message composition by providing orthographic and grammatical cues for effective word prediction.

By exploiting the current sentence context using the statistical techniques, a prediction system can provide more appropriate word choices to the user. The statistical information employed in any traditional word prediction system is mostly *n-gram* language models [1]. The *n-gram* word prediction model can be used to calculate the word frequency from the text corpus in *unigram*, *bigram* and *trigram* approaches in order to suggest the most probable next word. It also to be noted that in the case of *unigram* word prediction system, words are predicted by matching with the prefix of the current word entered so far only [2]. On the contraction *bigram* and *trigram* word prediction systems are taken consideration of one and two previous words along with the word prefix entered so far respectively [2]. It is also reported that trigram model gives better result in comparison of unigram and bigram [3].

However, the trigram model generally required massive amount of text corpus in compare with the *unigram* and *bigram* model [4]. As a consequence it slows down the prediction process due to searching the useful clues from the larger data set. Moreover, it is also quite possible that many of the *trigram* sequences may never occur even with the excessive amount of training data set [5], [6]. Here, in this research work we address this issue by using the popular *recommender algorithm*, *collaborative filtering technique* [7]. The *collaborative filtering algorithm* is implemented with the *Pearson correlation coefficient (PCC)* [8] to find the word similarity and the word similarities are used to predict the missing *bigram frequencies*. The approach is tested with different evaluation matrices like - percentage of key strokes required to predict the intended word, percentage of key strokes save by using the prediction engine, percentage of accuracy of our word prediction model. It is found that the propose model takes 15*:*2% and 17*:*4% less keystrokes in compare with the bigram predictor in respect to in-domain and out-domain data. The proposed model saves around 57% and the bigram model saves 41*:*8% key strokes for in-domain data. Similarly, for out-domain data the save percentage is 51*:*9 and 34*:*5 respectively. Around 93% and 91% of the words of the in-domain and out-domain data are accurately predicted by using the proposed system. However, on the same data set the bigram predictor accurately predicted around 87% band 83% of the word.

The rest of the paper is organized as follows. Section II discusses the prominent related works reported so far on the literatures. The proposed methodology is explained in Section III. The results and analysis of the proposed approach is presented in the Section IV. Finally, the paper is concluded with direction is future work in the Section V.

2. RELATED WORK

In recent section, we are going to discussed on several prominent works on word prediction strategies. It should be mention the scope of this section is bounded to the statistical word prediction only.

The statistical prediction systems mostly use *n-gram (uni-gram, bigram and trigram) language models* [1]. This model predicts the word on the basis of condition probability with the previous word. However, the *unigram model* uses only the word frequency information without including the previous word frequency. This is a very basic model generally used in the earlier word prediction system [2]. The advancement over the *unigram* is done by using *bigram and trigram model*. Both of the models are word frequency information along with the previous word frequency. The performance of *trigram model* generally better than the *bigram model* [9]. It is also to be mention that trigram model is usually considered as the baseline model in word prediction context [9]. However, the *data sparseness* is one of the major drawbacks of the *trigram model* [10], which can be address through *smoothing*. In spite of that another significant pitfall is the requirement of huge training data set to run the prediction system. Several popular works have done using the n gram language model which are discussed in the following.

Trnka et al. [4] compares three different text entry methods namely no prediction, basic word prediction and advanced word prediction. Word frequency model and trigram model is used in basic and advanced word prediction mechanism respectively. The experiment concludes that advanced word prediction can lead to higher text entry rates than basic and no prediction.

Arnott et al. have developed *Predictive Adaptive Lexicon (PAL)* [11] system. This is designed by considering the word frequency information and subtext which is being entered so far only without including the previous word. To enhance the word prediction capability, *PAL system* adds the new words automatically.

Carlberger et al. [12] have designed a statistical and adaptive system for predicting the next word in Swedish language known as *Profet*. The system uses the *bigram probability* to predict the probable word after completion of the current word and this current word is completed with the help of *unigram probability*. *Profet II* [13], the improved version of *Profet* includes *syntactic* and *semantic information* and to consider the previous last two words it used *Markov models*.

The Reactive Keyboard has been developed by Darragh et al. [14] for physically disable people. To predict the n^{th} character, the system uses the $(n-1)$ previously typed characters. This model stores the data using a tree structure and the context is reduced by one character if a match is not found.

Anson et al. [15] have been done one interesting empirical study on whether the word prediction or completion mechanisms would increase the typing speed for transcription based typing with an on-screen keyboard. Further, typing speed get more increased by integrating the word prediction technique over both the on-screen keyboard alone and the on-screen keyboard with word completion.

However, all the above mention works are not pay much attention of *data sparseness*. In this work we are given the most weatage on these aspects. As a consequence apart from *n-gram model*, we are using the *collaborative filtering approach* to provide the efficient prediction result over the previously used tradition approaches.

3. PROPOSED METHODOLOGY

It is established from the discussion of the previous section that the *data sparsity* is one of the major obstacle in the tradition word prediction mechanism (*n-gram model*). Here we propose an algorithm to overcome the effect of *data sparsity* in word prediction technique. In this approach we use item based *collaborative filtering* to estimate bigram frequency. This provides an alternate of *data smoothing* step in word prediction technique. The detail discussion is given in the following section.

Here, we take the *Wikipedia* english text [16] as the benchmark texts to train our system. It is also to be mention that the corpus have around 1:5 million of total words. The word-word matrix is created by removing the stop words from the considered corpus. So, the *cell* (*i; j*) of this matrix contains the frequency of the word *j* in the corpus which are occurred just after the word *i*. It is need to be mention that, around 20% of the cells contains null value due to *data sparsity* in the corpus. After calculating the word-word matrix, *Pearson Correlation Coefficient (PCC)* [8], [17] is employed to find similarity [18] between each i^{th} and j^{th} word by using Eqn 1. In Eqn 1, *Sim* (*i; j*) and W represents the similarity between i^{th} and j^{th} word and the subset of words which occurred with i^{th} and j^{th} respectively. Here, the frequency of word W with words *i* and the average frequency of word *i* are denoted by $f_{(w,i)}$ and $\overline{f}_1$ respectively.

$$\text{Sim}'(i,j) = \frac{\sum_{w \in W(i) \cap W(j)} (f_{w,j} - \overline{f}_i).(f_{w,j} - \overline{f}_j).}{\sqrt{\sum_{w \in W(i) \cap W(j)} (f_{w,j} - \overline{f}_i)^2}.\sqrt{\sum_{w \in W(i) \cap W(j)} (f_{w,j} - \overline{f}_j)}} \quad (2)$$

However PCC will overestimate the similarities between words who happen to have occurred with very few items in few contexts, but may not have similar in all contexts. To overcome this unexpected situation, we use the *Significance Weighting* approach proposed by Herlocker et al. [19], [20] over the previously used PCC approach. The *Significance Weighting* approach is defined in the Eqn 2, where $|W_i \cup W_j|$ is the number of words who have same frequencies with the i^{th} and j^{th} word and the threshold ä belongs to [0, 1]. In our experiment, we select 0.*2* as the value of ä according to cross validation. This algorithm generate a set S(*w*) of similar words for missing bigram frequency $f_{(w,i)}$according to Eqn 2 .

$$\text{Sim}'(i,j) = \frac{\text{Min}(|W_i \cup W_j|)}{\delta}.\text{Sim}(i,j) \quad (2)$$

Prediction of the missing frequencies is based on the similar words. Here we use *Similar Neighbors Selection* algorithm [21] with threshold (η) ranges in [0*;* 1]. In our selected dataset the η is 0.07 on the basis of cross validation. This algorithm generate a set of similar words S(*w*) for missing frequency $f_{(w,i)}$ according to the Eqn. 3.

$$S(w) = w_a \,|\mathrm{Sim}'(w_a, w) < \eta,\ w_a \neq w \tag{3}$$

The missing frequency can be estimated as illustrated in the Eqn. 4.

$$F(f_u, i) = \bar{f} + \frac{\sum_{i_k \in W(i)} \mathrm{Sim}'(i_k, i).(f_{u,i_k} - \bar{f}_k)}{\sum_{i_k \in W(i)} \mathrm{Sim}'(i_k, i)} \tag{4}$$

After the implementation of *Similar Neighbors Selection algorithm*, the approach is shifted towards the *bigram predictor* to predicting the intended word. The bigram predictor considers the just previous word along with the prefix of the word which is currently bening entered. The system first compares the existing prefix of the current word with all the words of the vocabulary and then it checks the *bigram probability* for each probable word which is calculated as $P(W_n \mid W_{(n-1)})$, where W_n is the probable or current word and $W_{(n-1)}$ is the previous word. It can be calculated for each probable word as shown in the Eqn 5, where *C* is the number of count of that particular word sequence.

$$P(W_n \mid W_{(n-1)}) = \frac{C(W_n W_{(n-1)})}{C(W_{(n-1)})} \tag{5}$$

4. RESULTS AND ANALYSIS

The above mention approach is implemented with the *"tm"* and *"recommederlab"* packages of R statistical tool on the above mention training data (Section III). The implemented system is tested on the selected testing modules which are different from training set. Here, we select both in-domain and out-domain testing data set with reference to training data. Three different in-domain data set are taken from the same source from where the training data are taken. However, each training set is contained the data set that are different from training set. Similarly, we have selected three different out-domain testing data set, which are taken from the selected portion of the English magazine *India Today* and the story book *The Da Vinci Code* written by Dan Brown. It also needs to be mention that each testing set have on an average around 100 words.

Here, we are going to evaluate the proposed system with reference of the following metrics - percentage of key strokes required to predict the intended word, percentage of key strokes save by using the prediction engine, percentage of accuracy of our word prediction model. To compare the efficacy of the proposed system, another similar *bigram predictor* is implemented with the same training data set. The both systems are evaluated on the same testing data with the same set of matrices as mention in the beginning of the current section.

Prior to discuss on the proposed system performance, we need to define the metrics on the basis of these the system performance will be evaluated. Number of keystroke required can refer the number of strokes on the keys of a keyboard are required to type the intended word. Let consider any word (W) has *n* number of characters. So, it requires *n* number of keystrokes to type W, while the system does not has any prediction mechanism. It is also to be mention that we are not considered any typographical or other error occurred during the typing. Otherwise it may require more than *n* number of keystrokes. On the other hand, if the system integrated with prediction mechanism, then it is required *m* number of keystrokes to type the same word *W* . Here, the *m* is less than or equal to *n*. More specially, if any character of the word W is predictable by the prediction engine, then *m* is less than *n*. Otherwise, *m* is equal to *n*. Suppose, the testing set contains *x* number of words each has $n_1, n_2, n_3 \ldots n_x$ number of characters respectively. The the total number of keystrokes required without incorporation of prediction model is $n_1 + n_2 + n_3 \ldots + n_x$. Hence, the original average required keystroke to type a word by the system which is not integrated with the prediction mechanism *acdeb* ($\overline{ks_{req-ori}}$) can be define as shown in the Eqn 6. On the contradiction, required total keystrokes due to the prediction model can be calculated as $m_1 + m_2 + m_3 + \ldots m_x$, where each *m*

is less than or equal to each respective *n*. So, the average required keystroke to type a word by the system incorporated with prediction model ($\overline{ks_{req-pre}}$) is define in the Eqn 7.

$$(\overline{ks_{req-pre}}) = \frac{\sum_{i=1}^{x} n_i}{x} \tag{6}$$

$$(\overline{ks_{req-pre}}) = \frac{\sum_{i=1}^{x} m_i}{x}; m_i \le n_i, \forall i = 1 \cdots n \tag{7}$$

The percentage of key stroke required (Per – ks_{req}) can be calculated by dividing ($\overline{ks_{req-pre}}$) by the ($\overline{ks_{req-ori}}$) and multiplying it with hundred as shown in Eqn 8.

$$\text{Per} - ks_{req} = \frac{\overline{ks_{req-pre}}}{ks_{req-ori}} \times 100 \tag{8}$$

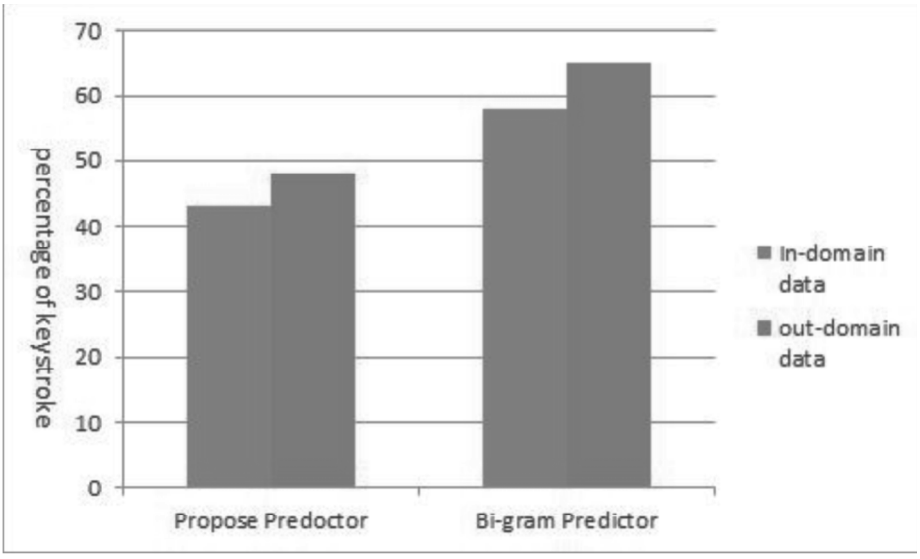

Fig. 1. Percentage of keystrokes required for proposed system and bigram predictor

The Fig. 1 is illustrated the percentage of keystroke is required to predict the intended word both for our proposed system as well as the *bigram predictor*, which is calculated by using the Eqn 8. The figure conveys that both of the considered systems are required less number of keystroke to type the intended word, for in-domain data in comparison with out domain-data. The proposed system is required on an average 5:1% less number of keystroke while typing in-domain data in compare with the out-domain data. Similarly, the *bigram system* takes around 7:3% less number of keystroke for in-domain data in reference without-domain data typing. It is also conveying that our proposed system takes on an average 15:2% and 17:4% less keystroke than normal *bigram predictor* in respect to in-domain and out-domain data correspondingly.

The above discussed experimental result can easily define the another considered matric, percentage of key stroke (Per – ks_{save}) save It can be define by subtracting the percentage of key stroke required (Per – ks_{req}) matric from hundred as shown in Eqn. 9. The Fig. IV is illustrated the percentage of key stroke save over the *bigram predictor* due to use of our system. It shown that around 57% and 41:8% percentages of key strokes save in proposed and *bigram predictor* for in-domain

Similarly, for the out domain data the system are saved on an average 51.9% and 34.5% of key stroke.

$$\text{Per} - ks_{save} = 100 - \text{Avg} - ks_{req} \tag{9}$$

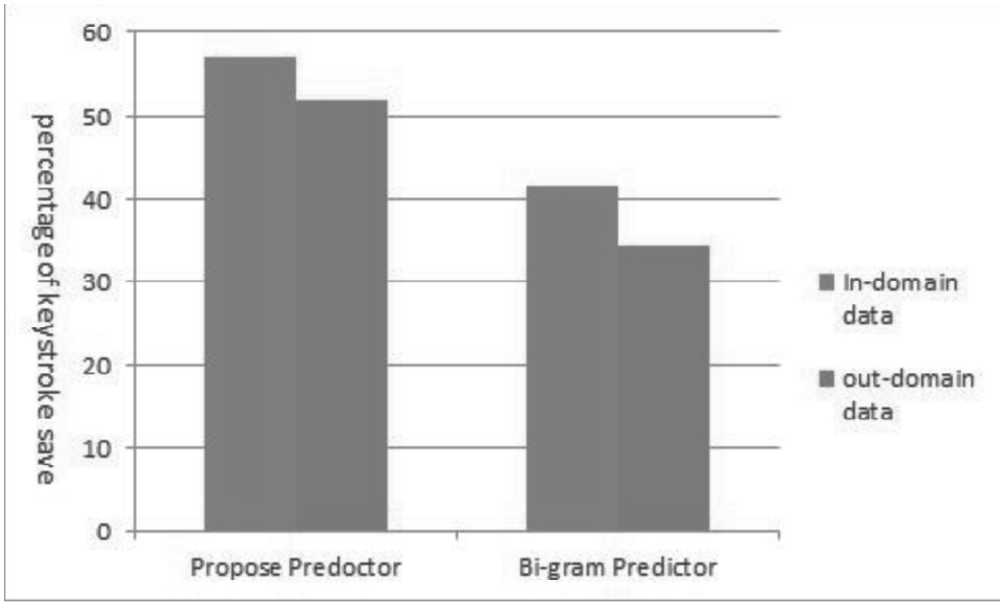

Fig. 2. Percentage of keystrokes saves on proposed system and bigram predictor

Here the metric percentage of accuracy is referred to whether any particular intended word is predicted before it typed completely. Hence, the number of keystrokes are required are less for typed any particular word using the prediction system in compare to without of that. On the contradiction, if it requires equal number of keystrokes than the word is unpredictable. Suppose, the testing set consists of p number of words and the system able to predict q number of words of that set. Then, the percentage of accurately predictable word (Pre – predic) is calculated by dividing q by p and multiplying it by hundred as shown in 10. So, the percentage of inaccurately predicted or unpredictable word can be calculated by subtracting (Pre – predic) the from hundred as shown in the Eqn. 11. The same matric can also be calculated by divide the subtract q from p by p and multiplied it by 100 as shown in the Eqn. 12.

$$\text{Per} - acc = (q/p) \times 100 \tag{10}$$

$$\text{Per} - acc = 100 - \text{Per} - \text{predic} \tag{11}$$

$$\text{Per} - acc = (p - q)/p \times 100 \tag{12}$$

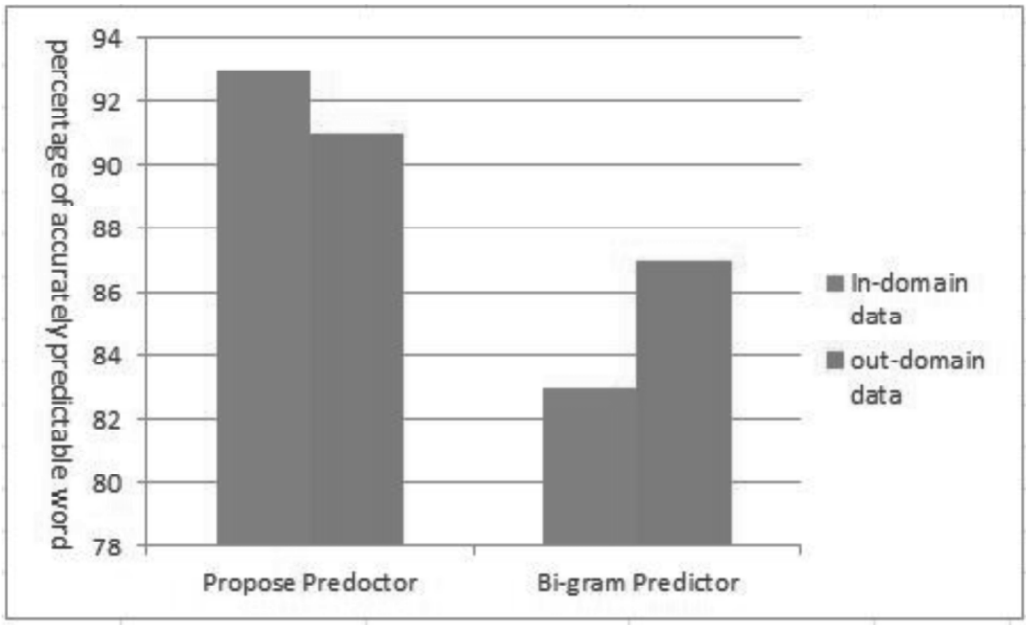

Fig. 3. Percentage of accurately predictable word in proposed system and bigram predictor

The test result obtained from our experimental setup in the basis of and matrices detected the Fig. 3. It is conveyed that around 93% and 91% of the words of the in-domain and out-domain data set are accurately predictable by our system. So, obviously unpredictable word percentage is 7% and 9% respectively on the same data set. However, it is shown that around 87% and 83% of words are accurately predictable, if we use *bigram predictor* for the same test scenario. Hence, the percentage of unpredictable word is 13% and 17% of in-domain and out-domain data set respectively.

5. CONCLUSION

Predicting the most probable next word is one of the foremost way out to enhance the text entry rate which is driven through saving the number of required keystrokes. However, the performance of the word prediction systems developed by using the traditional approaches is mainly confined due the *data sparsity* in their training set. To overcome this *data sparsity* problem, we develop a word prediction system using *collaborative filtering* approach which is initially implemented by *Pearson correlation coefficient (PCC)*. However, it is overestimated the similarity between words. This unexpected situation is taken care by *Significance Weighting* approach over the *PCC* method. Nevertheless, *Significance Weighting* approach suffers from missing frequencies problem which is resolved by employing *Similar Neighbors Selection* algorithm. Finally, the *data sparsity problem* is resolved by following all this previous mention approaches. After that the approach is shifted towards the *bigram predictor* to develop the final system. The develop system is critically evaluated through different considered matrices in reference with both for in-domain and out-domain testing data. The experimental outcome established that our develop word prediction system is outperformed than the simple *bigram predictor*. However, here we only restrict up to *bigram predictor only*. In future this limitation would be overcome with the *trigram predictor* integrated with *collaborative filtering approach*.

6. REFERENCES

1. Wikipedia, "n-gram," https://en.wikipedia.org/wiki/Ngram, accessed on April 2015.
2. A. Fazly, "The Use of Syntax in Word Completion Utilities," Master's thesis, Department of Computer Science, University of Toronto, 2002.
3. F. Kishino, Y. Kitamura, H. Kato, and N. Nagata, *Entertainment Computing-ICEC 2005: 4th International Conference, Sanda, Japan, September 19-21, 2005, Proceedings*. Springer, 2005, vol. 3711.
4. K. Trnka, J. McCaw, D. Yarrington, K. F. McCoy, and C. Pennington, "User Interaction with Word Prediction: The Effects of Prediction Quality," *ACM Transaction on Accessible Computing*, vol. 1, no. 3, p. 134, 2009.
5. S. M. Katz, "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer," *IEEE Transactions on Acoustics, Speech and Signal Processing*, pp. 400–401, 1987.
6. H. Ma, I. King, and M. Lyu, "Effective missing data prediction for collaborative filtering," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 39–46.
7. J. B. Schafer, D. Frankowski, and J. H. S. Sen, "Collaborative Filtering Recommender Systems," in *The adaptive web*. Springer Berlin Heidelberg, 2007, pp. 291–324.
8. U. Shardanand and P. Maes, "Social information filtering: algorithms for automating word of mouth," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press/Addison-Wesley Publishing Co, 1998, pp. 210–217.
9. D. Jurafsky and J. Martin, *Speech and Language Processing*. Pearson Education India, 2000.
10. G. Grefenstette and S. Renals, "Text-and speech-triggered information access: Introduction," in *Text-and Speech-Triggered Information Access*. Springer Berlin Heidelberg, 2003, pp. 1–5.
11. J. L. Arnott, A. F. Newell, and N. Alm, "Prediction and conversational momentum in an augmentative communication system," *Communications of the ACM*, vol. 35(5), p. 4657, 1992.

12. A. Carlberger, T. Magnuson, J. Carlberger, H. Wachtmeister, and S. Hunnicutt, "Probability-BasedWord Prediction forWriting Support in Dyslexia," in *Proceedings of Fonetik Conference*, 1997, p. 1720.

13. J. Carlberger, "Design and Implementation of a Probabilistic Word Prediciton Program," Master's thesis, Computer Science, Nada, KTH, Stockholm (Sweden), 1997.

14. J. J. Darragh and I. H. Witten, "Adaptive Predictive Text Generation and the Reactive Keyboard," *Interacting with Computers*, vol. 3(1), p. 2750, 1991.

15. D. Anson, P. Moist, M. Przywara, H. Wells, H. Saylor, and H. Maxime, "The Effects of Word Completion and Word Prediction on Typing Rates using On-screen Keyboards," in *Assistive technology: the official journal of RESNA*, vol. 18(2), 2006, p. 146 154.

16. "English Wikipedia," https://en.wikipedia.org/wiki/English Wikipedia, accessed on January, 2015.

17. J. Benesty, J. Chen, Y. Huang, and I. Cohen, *Topics in Signal Processing*. Springer, 2009, vol. 2, ch. Pearson Correlation Coefficient, pp. 1–4.

18. S. Badrul, G. Karypis, J. Konstan, and J. Riedl, " Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 285 – 295.

19. J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1999, pp. 230 – 237.

20. J. Herlocker, J. A. Konstan, and J. Riedl, " An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms," *Information retrieval*, vol. 5(4), pp. 287 – 310, 2002.

21. M. Hao, I. King, and M. R. Lyu, "Effective missing data prediction for collaborative filtering," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 39 – 46.