Blind Recognition of Text Input on Mobile Devices via Natural Language Processing

Qinggang Yue
UMass Lowell, USA
qye@cs.uml.edu

Zhen Ling
Southeast University, China
zhenling@seu.edu.cn

Wei Yu
Towson University, USA
wyu@towson.edu

Benyuan Liu
UMass Lowell, USA
bliu@cs.uml.edu

Xinwen Fu
UMass Lowell, USA
xinwenfu@cs.uml.edu

ABSTRACT

In this paper, we investigate how to retrieve meaningful English text input on mobile devices from recorded videos while the text is illegible in the videos. In our previous work, we were able to retrieve random passwords with high success rate at a certain distance. When the distance increases, the success rate of recovering passwords decreases. However, if the input is meaningful text such as email messages, we can further increase the success rate via natural language processing techniques since the text follows spelling and grammar rules and is context sensitive. The process of retrieving the text from videos can be modeled as noisy channels. We first derive candidate words for each word of the input sentence, model the whole sentence with a Hidden Markov model and then apply the trigram language model to derive the original sentence. Our experiments validate our technique of retrieving meaningful English text input on mobile devices from recorded videos.

Categories and Subject Descriptors

K.4.1 [**COMPUTERS AND SOCIETY**]: Public Policy Issues—*Privacy*

General Terms

Human Factors, Security

Keywords

Mobile Security, Computer vision, Natural Language Processing

1. INTRODUCTION

In this paper, we blindly retrieve English sentences such as email messages entered on a mobile device. In our previous work [20], we demonstrated how to blindly retrieve a password from a recorded video. The same theory in [20] can be used to recover texts. However, since messages such as an email message follow a language model, we can utilize natural language processing techniques (NLP) to correct the recovered sentences and improve the accuracy. We show that although recovered messages via our technique in [20] are illegible at first, our novel NLP based techniques can accurately correct those messages.

PAMCO'15, June 22-25, 2015, Hangzhou, China

DOI: http://dx.doi.org/10.1145/2757302.2757304

Figure 1 shows the basic idea of the technique in our previous work [20] recognizing passwords entered on a tablet. In the video frame, the fingertip in the solid edge green bounding box touches the screen. This frame is called touching frame. If we can obtain the touched point where the fingertip touches on the screen and derive the homography relation [16] between the keyboard in the touching frame and the reference keyboard in Figure 2, we can derive the touched key by mapping the touched point to the reference keyboard. In this example, the mapped point falls into the area of the key "U". Therefore, "U" is the touch-input.

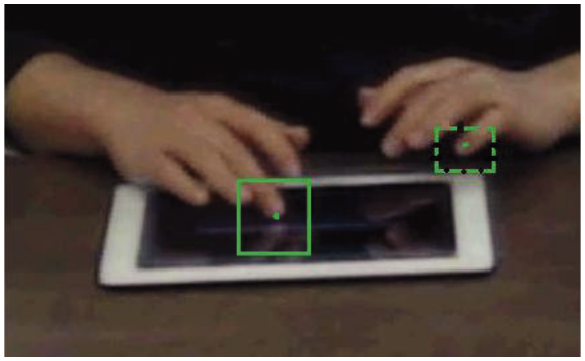

Figure 1: Touching Frame (Zoomed in)

Figure 2: Reference Software Keyboard

The technique in [20] can be used to recover meaningful text entered on a mobile device. However, then the distance between the camera and target device increases, the recovered text will illegible. Our problem of retrieving the text can be treated as a spelling correction problem. The text actually goes through two noisy channels: the noisy touch input channel which models the user touching/typing process, and the noisy reconstruction channel which models the process of recovering the text via the technique in [20]. The reconstruction channel introduce the dominant errors, which are caused by the inaccurate recognition of touched points in

the touch frames by the applied computer vision techniques. This makes our problem much different from the traditional spelling correction problem.

We adopt two major steps to correct the raw recovered text. We first apply the unigram language model to correct each word of the input sentence independently. We consider the possible types of errors and build the editable word graph model to derive word candidates for each word. Each candidate is scored based on our error models. Therefore, we can select the candidate with the highest score as the intended input. Second, we apply the n-gram language model, particularly the trigram model, to perform further correction. Words in a sentence are not isolated, follows the grammar rules and has its context. Such information can be utilized for the correction. We model the sentence as a graph based on the Hidden Markov Model and apply the the n-gram language model to derive the most possible sentences.

The rest of this paper is organized as follows. Section 2 gives the basic idea of retrieving and correcting the text input. Section 3 introduces how to correct the isolated words. Section 4 shows how to utilize the context information to improve the results from Section 1. Evaluation is given in Section 5. In Section 6, we introduce the most related work. Section 7 discusses the future work. Section 8 concludes this paper.

2. BASIC IDEA

Figure 3 shows the steps of recognizing text entered on mobile devices. A user touch inputs sentences on a mobile device. Since the user may make mistakes when typing, we model the process of touch-inputting as a noisy *touch input channel*. After getting the video, we apply the reconstructing technique in [20] and derive a sequence of characters $c_1, c_2 \dots c_k$. Since the reconstruction process may recognize the entered characters wrong, we model this process as a noisy *reconstruction channel*. Therefore, the two noisy channels, touch input channel and reconstruction channel, have a combined effect on the recognition result of individual characters. Our previous work [20] stops here.

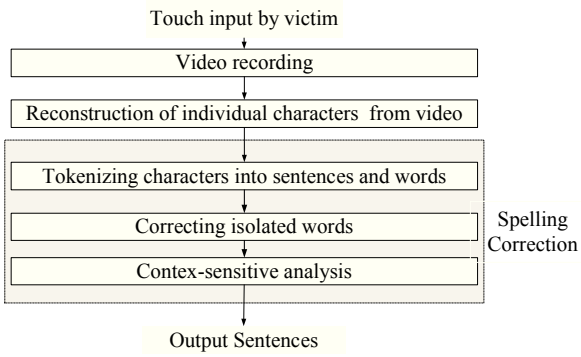

Figure 3: Reconstructing Text Input

We can actually correct the sequence of characters $c_1, c_2 \dots c_k$ by applying natural language processing techniques given that sentences follow spelling and syntactic rules. We first tokenize the character sequence $c_1, c_2 \dots c_k$ into sentences. In this paper, we assume we are able to detect the start and end of sentences correctly. This assumption is similar to the assumption in [22]. For each sentence, we tokenize it into a word sequence $o_1, o_2 \dots o_n$, where n may be different for different sentences. We build models for the touch input channel and reconstruction channel and consider the combined effect of the two noisy channels in order to correct isolated words.We score each candidate word. Words with the highest scores hopefully form the original sentence. Context-sensitive analysis is further applied, considering the context and syntax of neighbouring words. This will further improve the result.

3. CORRECTING ISOLATED WORDS

In this section, we introduce how to derive individual words using unigram language models. A raw word can be reconstructed via our technique in [20]. However, the touch-inputting and the reconstruction processes may introduce errors and behave as noisy channels, denoted as the touch input channel and reconstruction channel respectively. To generate word candidates, we build the editable word graph model from the recovered characters. We filter these candidates through a dictionary to derive possible words while discarding non-word candidates. We score these word candidates based on our error models.

3.1 Overview

Figure 4 re-interprets Figure 3 and introduces the steps of recognizing individual words with the unigram language model. The intended word is w. w goes through the touch input channel and the output could be different from w because of human input errors. The touch input process is recorded by attackers, who apply computer vision analysis and try to reconstruct the input. The reconstruction process introduces errors because of the applied computer vision techniques and the output of the reconstruction channel is o. Based on the models of the touch input channel and recognition channel, we design algorithms and generate a list of word candidates, $w_1, w_2, \dots$. We design metrics, score each candidate and have the list of word candidates and their associated scores, $\{w_1, p_1\}, \{w_2, p_2\}, \dots$. Given the list, we may pick up the word candidate with the highest score as the intended word w. As introduced in Section 4, we may further improve the recognition accuracy by performing the context-sensitive analysis via the n-gram language model.

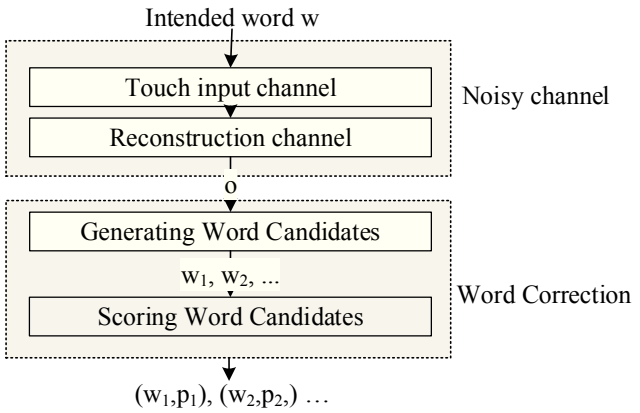

Figure 4: Correcting Words by Unigram Language Model

Therefore our problem can be formalized as: given the reconstructed word o, find the word w in the vocabulary V, that maximize the probability of $P(w|o)$:

$$w = \underset{w \in V}{\operatorname{argmax}} P(w|o), \tag{1}$$

Applying the Bayes' theory, we get

$$\begin{aligned} w &= \underset{w \in V}{\operatorname{argmax}} P(w|o) \\ &= \underset{w \in V}{\operatorname{argmax}} \frac{P(o|w)P(w)}{P(s)} \\ &= \underset{w \in V}{\operatorname{argmax}} P(o|w)P(w) \end{aligned} \tag{2}$$

$P(o|w)$ is *the error model or the channel model* and it models the probability that the intended word w is changed to o through the noisy channel. For example, the user may intend to type the word "building", but the word "bukldijg" is typed. Even worse, after going through the computer vision analysis part, the word "ukldkjy" is reconstructed. $P(w)$ is the source model or the language model. It models the probability the word w appears in a particular language. For example, even though the word "for" and "fur" are both correct words, "for" is more likely to be typed than the word "fur".

In the following subsections, the error model will be first introduced: we show how to generate the word candidates and score them with probabilities by analyzing and modeling the above two noisy channels. Then we introduce the language model. With the channel models and the source model, we derive the score of word candidate w as $Score(w) = P(o|w)P(w)$. The candidate word with the highest score may be the most possible intended word if the uni-gram language model is used. In Section 4 we introduce how to further improve the correction with context-sensitive analysis.

3.2 Touch Input Channel Model

People may make mistakes when typing the words. For the common spelling correction problems, this is usually modeled by the insertion, deletion, replacing, or reversing operations [12], and these spelling errors are introduced while people type on the hard keyboard. People may also make mistakes when touch inputting. Since the keys on the touch screen are small, fingers may touch the wrong keys. If a person touch inputs with both hands, the error model of the touch input channel is similar to the error model of the typing channel. If a person touch-inputs with one finger on small mobile devices like smartphones, we have only the replacing errors.

3.3 Reconstruction Channel Model

Recall that to recognize a character from a recorded video, we detect the touching fingertip, locate the touched point and map the touched point to the reference image. The challenge is that locating the touched point often introduces errors because of lighting, camera resolution and other issues.

Character candidate model: It can be observed from Figure 5 that if we can detect the fingertip correctly, the touched point should be in the area confined by the green rectangle, which is generated by our computer vision technique locating the fingertip. Correspondingly, if we map this area to the reference image, we can infer the character area as shown in Figure 6. Therefore, we propose the geometry based *character candidate model*: even if the computer vision analysis of touched points makes mistakes, the character candidates should be keys in this character areas if the touching fingertip is detected correctly.

3.4 Generating Word Candidates

Word Graph Model Given the candidates of each character, we can further form the words. If we assume that all the touching

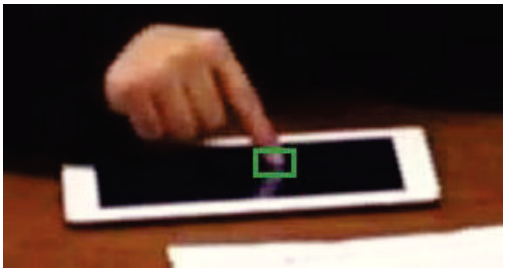

Figure 5: Larger Touched Area

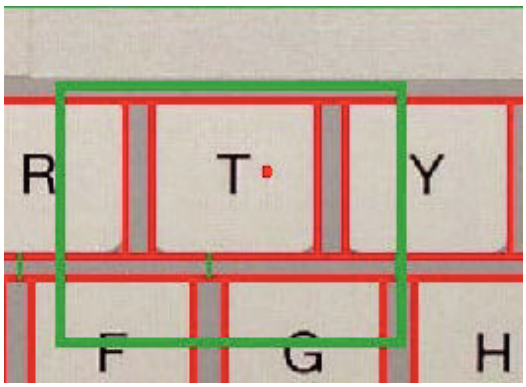

Figure 6: Character Candidate Model - Possible Touched Keys

actions are correctly detected and the touching fingertip are accurately located. The word candidates can be derived from combining all the characters. This combination process can be modeled as a directed acyclic graph with the character candidates as vertex and the links between subsequent characters as edges. For example, in one of our experiments the touched word "state" is reconstructed as "xtzte". We derive the character candidates of all the characters, and build the word graph model for the word "xtzte" shown by Figure 7. From this figure, it is clear that the red path indicates the correct word "state".

Editable Word Graph Model: The word graph model models the case that there is no errors while detecting touching actions and fingertip. However, some touching actions can be missed through our analysis while some non-touching actions are wrongly detected as touching actions. The touching fingertip can also be located wrong. To model such kind of errors, we perform edition to the word graph model: *inserting* models the non-detected touching frames, *deleting* models detecting the non-touching actions as touching actions, and *replacing* models the error of mapping the touched key to a

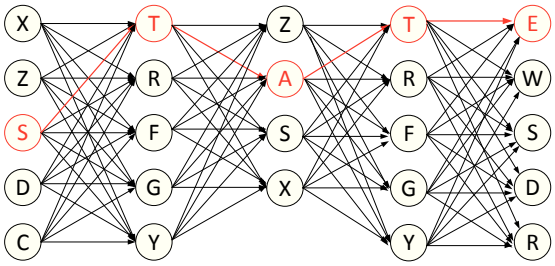

Figure 7: Word Graph Model

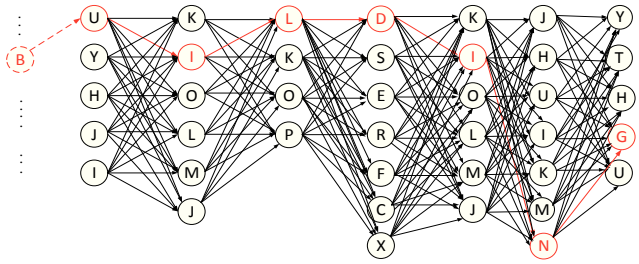

Figure 8: Editable Word Graph Model

complete wrong key. This revised model is called the *Editable Word Graph Model*. We can also observe this editable word graph model actually also includes errors introduced by the touch input channel, which introduces similar errors as the reconstruction channel.

Figure 8 is an example from our experiments. The user intends to type the word "building", and indeed typed the correct word. However, the computer vision analysis reconstructed the word as "ukldkjy". After applying the character model and word graph model, we get the graph indicated by the solid lines in Figure 8. It is clear that the graph path by the red solid lines compose the word "uilding", after the inserting the character 'b' in the first place of the graph word candidate, presented by the dash line, we get the word candidate "building". Other editing operations are performed in a similar way. To limit the number of word candidates, we set our edit distance to be 1. That is, at most one character may be inserted, deleted, or replaced,

Dictionary Filtering: The combinations of the characters along the path of the graph are the possible word candidates. It is obvious that some candidates are not correct words. Thus, a dictionary is deployed to filter the non-words. In this paper, we use the medium sized Spell Checking Oriented Word Lists (SCOWL) [1], which contains 101,895 words. These words are the common words from several dictionaries.

3.5 Scoring Word Candidates

Given the word candidates, we need to score them. We first score the word graph model candidates. The intuition for scoring the candidates is that the smaller edit distance between the reconstructed word and the word candidate, the higher score the candidate should have. We first define the distance between all the corresponding characters of the candidate and the reconstructed word, and multiply these distances to derive the distance of the word candidates. To get the distance between characters, we first define the coordinates of the keys on the keyboard, (r, c) where r is the row number and c column number. The distances between the character C_1 with coordinate (r_1, c_1) and the character $C_2(r_2, c_2)$ is $(|r_1 - r_2 + 1| \times |c_1 - c_2 + 1|)$. Getting the distance between corresponding characters, we derive the score of the word candidate by multiplying the character distances together.

Then, we normalize the scores into the probability. The smaller distance from the reconstructed word to a word candidate, the larger probability this candidate as the original word. We first derive the inverse of the scores, and then normalize all the scores divided by the sum of all the inverses. The final scores form a probability distribution, where every candidate has a higher than zero probability and the sum of probabilities of all the candidate words is 1.

For the editable word graph model, since the detection errors from the computer vision analysis are random, we assign the insertion or deletion of a character with a probability of 1/26 since there are 26 characters. For the replacing operation, for different characters we assign different probabilities according to their geometrical distance on the keyboard. The probabilities are generated by first calculating the character distances, then do the inverse and normalization to make the scores form a probability distribution. The same score is applied to the replacing operation for the touch input channel.

3.6 Source Model

Source model models the probabilities of words appearing in the language. For different languages, the source model should be generated in different ways considering the nature of the language [14]. For English, it is good enough to generate the model by counting the appearance of each word from a particular training corpus.

In this paper, we build the source model from the well-known British National Corpus (BNC) [4] by counting the appearance of each word and apply the Maximum Likelihood Estimation algorithm. BNC has 100 million words containing both spoken and written English from various sources and scenarios.

4. CONTEXT-SENSITIVE ANALYSIS

In Section 3, we introduced how to get word candidates for each word associated with scores indicating their probabilities being the intended word. We now show that we can actually apply the trigram language models considering the context of the words (the neighboring words should be combined into a meaningful sentence) and further improve the recognition of text inputs on mobile devices.

There are two types of spelling errors: non-word errors and real word errors. To deal with real word errors, context-sensitive error detection and correction algorithms have been proposed. There are two major types [17] [6]: algorithms based on language syntactic information [13] [8] [18] [5] and algorithms based on language semantic information [15] [10]. The language syntactic information based algorithms apply the information of the probability of the sentences, word ngram language models, or POS tagging [8] and others for the analysis in the context. The algorithms based on semantic information utilizes the similarity between the semantics of words and the context.

In our scenario, we may have many different candidates for every word. In the conventional spelling correction problems, there are not so many errors in one sentence. The semantic based algorithms are not directly applicable to our scenarios. In our case, the context of the word is not determined since every word has some candidates in our case. However, we can still apply the n-gram languages to select the combinations and the word candidates since the combinations with a higher probability would be more possible to be the correct word. This matches the principle of the language model.

5. EVALUATION

We use iPhone to perform experiments and validate the attack and randomly selected one paragraph from Wall Street Journal and the example sentences from [19] as original sentences, as shown in Table 1. Our experiments show that from 3 meters or 4 meters we can derive the inputs correctly. From 5 meters, we may not be able to derive a completely right sentence and some words may be wrong. The reason is that there are too many wrong words from the reconstruction process.

Original Sentence	Reconstructed Sentence	Noisy Channel Result	Final Result
when your round is a short one you take a walk	whdn your rounc ix z wnpft ljd you tzkd z szl,	when your round is a short one you take a walk	when your round is a short one you take a walk
when it is a long one you take a cab	shem it ix q lony one you tzkd z cab	when it is a long one you take a **can**	when it is a long one you take a cab
if you know your enemy and you know yourself you need not fear the results of a hundred battles	if uou ojkw your emeky amd yoi kmow yourself uou need not fear the redultd ot a jundred battles	**of** you **like** your enemy and you know yourself you need not fear the results of a hundred battles	if you **like** your enemy and you know yourself you need not fear the results of a hundred battles
if you know neither the enemy nor yourself you will succumb in every battle	if you knoa heither thd ensmy nor ykurself you aill xuccumb in dgery battle	**of** you know neither the enemy **not** yourself you will succumb in every battle	if you know **brother** the enemy **not** yourself you will succumb in every battle
i plan to stay at home	i p.an to ztay zt home	i plan to stay at home	i plan to stay at home
i am busy tonight	k zm buxy tonight	i **an** busy tonight	i am busy tonight
the rules of conduct released by city, state and county officials are based on requests from protesters and agreed upon by law enforcement after weeks of discussions aimed at building a relationship between protest leaders and police	the rulex of cojduct releaaed by city , xtzte anx county ovficials are bzxed oj reauexts ffom protesters and agreed upon by law enforcement after weeks of dizcuxzionx aiked zt building a relztiojship betweej protext leacers and police	the rules of conduct released by city, state and county officials are based **in** requests from protesters and agreed upon by law enforcement after weeks of discussions aimed at building a relationship between protect leaders and police	the rules of conduct released by city, state and county officials are based on requests from protesters and agreed upon by law enforcement after weeks of discussions aimed at building a relationship between protect leaders and police

Table 1: Example Reconstruction Results

Table 1 gives some example results, including the intermediate results and the final result after the context-sensitive analysis is performed. The bold words in the last two columns refer to words that are not correctly derived.

6. RELATED WORK

This section introduces the most related work on inferring the sentences from the character sequences derived from various side channels. Xu et al. [19] retrieved the touch input on touch devices from the videos taken from some distance. They first train a key classifier based on the appearance of the screen and the fingertip when a key is touched. The classifier is applied to recognize the touched keys. Then, they apply one edit distance model to derive the words from the original string and build a unigram model trained on the Brown corpus [7] to select the correct word from the candidates. Obviously, this work can be improved by further applying bigram or trigram models to consider the language context. Besides, the Brown Corpus is too small for training a language model which contains only about 1 million words.

Balzarotti et al. [2] focused on retrieving the typing inputs from the video of people typing the hard keyboard taken with the camera directly upon the keyboard. For every frame in the video, they deploy lighting features to derive the touched and the non-touched key group. Then for consecutive frames they group them into different key candidates. Given the character candidates, they build a word model and apply the noisy channel model to get scores for different words. In the final step, they apply the 3-gram and 4-gram analysis to select the word candidates for the sentence according to the frequency of the ngram tuples. For the non-existing ngram tuple, the related word candidates are discarded and the remained candidates are re-ordered. The problems is even if a large corpus [3] is deployed, there are still possible trigrams that do not appear in it [11]. From their evaluation, they can retrieve 64% words correctly for the top 5 candidates. But from the example they give, the resultant sentence is not quite readable since 5 candidates for each word are still too many.

Zhuang et al. [21, 22] analyzed the problem of hard keyboard acoustic emanations. They first train a character classification confusion matrix, which models the probability that one character is misclassified as another character. The conditional probability of retrieved word given the dictionary word is derived by multiplying the corresponding value in the confusion matrix. After getting the word candidates, they apply the trigram language model with the Hidden Markov Model to derive the target sentence. But, they did not mention what corpus was used to train the language model and how they deal with the non-existing trigram probabilities. They also assume to be able to get the correct length of the words.

7. FUTURE WORK

We plan to improve our error model by analyzing the text we get from the user input on the touch screen rather than the edit distance model we use now in Section 3. We plan to get a corpus of spelling errors, then do a statistical analysis of the probability of the edit distance operations among all the characters.

Another possible future work is about inserting or replacing the space keys. Currently, we do this manually if we find that some space keys are detected wrong. Words are tokenized by the space key. If the space key between two words is missed during the analysis, we need to insert one space key to correctly analyze the sentence. To address the issue of recognizing space keys, our computer vision techniques have to be improved to deal with those cases that keys are located at the rear part of the keyboard. If the space keys can still not be recognized automatically, we have to address the problem of where to insert the space key and how many spaces keys should be inserted. We plan to apply the unsupervised learning process [9] for word segmentation to address this challenge.

In this paper, similar to other related papers, smart words (such as abbreviations, new words on the social network) are not analyzed, neither the special keys such as the backspace key and the shift key. The challenge is the case that we make mistakes when retrieving those special keys. For the touch enabled devices, a user may alternate the keyboard by touching those special keys. If we miss those special keys, the results could be very confusing. Our future works should consider such cases.

To reduce the time and space complexity to store and retrieval the words, we could consider constructing a trie-structure. A trie models the dictionary as a tree. Depth first search can be applied to check whether a word candidate exists in the dictionary. This will reduce both the computing and memory complexity compared with using a simple dictionary.

8. CONCLUSION

In this paper, we explore the n-gram language models to recognize text inputs on touch-enabled devices, given the input is meaningful English sentences and has its grammars and semantics. We model the human touch input process and character reconstruction process from a video as a noisy touch input channel and reconstruction channel respectively. These two noisy channel introduce errors to the recovered words. Characters of those words may be inserted, deleted, replaced or reversed. We developed the editable graph word model in order to generate the word candidates and score them based on the error models. To further improve the accuracy of recognizing the meaningful text input, we consider the context information in our analysis. We build n-gram language models from the British National Corpus and apply them to analyze sentences via the hidden markov model. Our experiment results validate the attack of blind recognition of text input on mobile devices.

9. ACKNOWLEDGEMENTS

This work is supported in part by National Natural Science Foundation of China under grants 61272054 and 61402104. Jiangsu Provincial Key Laboratory of Network and Information Security under grants BM2003201, Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under grants 93K-9, US NSF grants 1350145, 1117175, 0953620 and 1116644. Any opinions, findings, conclusions, and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

10. REFERENCES

[1] K. Atkinson. Spell checking oriented word lists (scowl), Feb 2015. http://wordlist.aspell.net/.

[2] D. Balzarotti, M. Cova, and G. Vigna. Clearshot: Eavesdropping on keyboard input from video. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, SP '08, pages 170–183, 2008.

[3] T. Brants and A. Franz. Web 1t 5-gram version 1 ldc2006t13. Philadelphia: Linguistic Data Consortium, 2006.

[4] L. Burnard. Reference guide for the british national corpus. Technical report, Oxford University Computing Services, 2000.

[5] K. Church and W. Gale. Probability scoring for spelling correction. *Statistics and Computing*, 1:93–103, 1991.

[6] D. Fossati and B. D. Eugenio. I saw tree trees in the park: How to correct real-word spelling mistakes. In *In Proceedings of the 6th International Conference on Language Resources and Evaluation*, pages 896–901, 2008.

[7] W. N. Francis and H. Kucera. Brown corpus manual. Brown University, 1979.

[8] A. R. Golding and Y. Schabes. Combining trigram-based and feature-based methods for context-sensitive spelling correction. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pages 71–78, 1996.

[9] J. Heymann, O. Walter, R. Haeb-Umbach, and B. Raj. Unsupervised word segmentation from noisy input. In *Automatic Speech Recognition and Understanding Workshop (ASRU 2013)*, Dec. 2013.

[10] G. Hirst and A. Budanitsky. Correcting real-word spelling errors by restoring lexical cohesion. Department of Computer Science, Toronto, 2001.

[11] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

[12] M. D. Kernighan, K. W. Church, and W. A. Gale. A spelling correction program based on a noisy channel model. In *Proceedings of the 13th Conference on Computational Linguistics - Volume 2*, COLING '90, pages 205–210. Association for Computational Linguistics, 1990.

[13] E. Mays, F. J. Damerau, and R. L. Mercer. Context based spelling correction. *Inf. Process. Manage.*, 27(5):517–522, 1991.

[14] T. A. Pirinen and S. Hardwick. Effect of language and error models on efficiency of finite-state spell-checking and correction*, 2012.

[15] D. St-Onge. Detecting and Correcting Malapropisms with Lexical Chains. Master's thesis, Department of Computer Science, University of Toronto, 1995.

[16] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[17] S. Verberne. Context-sensitive spell checking based on word trigram probabilities. Master's thesis, University of Nijmegen, 2002.

[18] A. L. Wilcox-O'Hearn. *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, chapter A Noisy Channel Model Framework for Grammatical Correction, pages 109–114. Association for Computational Linguistics, 2013.

[19] Y. Xu, J. Heinly, A. M. White, F. Monrose, and J.-M. Frahm. Seeing double: Reconstructing obscured typed input from repeated compromising reflections. In *Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS)*, 2013.

[20] Q. Yue, Z. Ling, X. Fu, B. Liu, K. Ren, and W. Zhao. Blind recognition of touched keys on mobile devices. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 1403–1414. ACM, 2014.

[21] L. Zhuang, F. Zhou, and J. D. Tygar. Keyboard acoustic emanations revisited. In *In Proceedings of the 12th ACM Conference on Computer and Communications Security*, 2005.

[22] L. Zhuang, F. Zhou, and J. D. Tygar. Keyboard acoustic emanations revisited. *ACM Trans. Inf. Syst. Secur.*, 13(1):3:1–3:26, Nov. 2009.