Ranking Model for Improving Input Method Accuracy Based on LambdaMART

Qiaofei Wang, Ting Li, Gang Qiao, Jianmin Wu, Tianhuang Su

GBU, *BaiduInc*, *China*, 518052
E-mail: {wangqiaofei01, liting21, qiaogang01, wujianmin01, sudianhuang}@baidu.com

1 Introduction

In an age of information explosion, mobile devices gradually become a bridge linking individuals and the outside world. As [1] reveals, there are 5.11 billion unique mobile users in the world today. Along with the increasing demand of sharing information through mobile devices, people have higher requirements on the accuracy of the input method, which is an essential tool for human-computer interaction.

However, the input experience of common input methods is barely satisfactory as the size of keyboard is generally limited. According to our statistics, there are about 20% misspelled words in the process of inputting English in mobile devices with touch screen. Besides, as the expected word does not always appear in the first position the input method displays, additional operations like scrolling the list of candidate words are required occasionally, which make the input progress more inefficient. Therefore, enhancing the accuracy of error correction and word prediction is of great importance for input methods.

Figure 1: An example of error correction in Facemoji keyboard

The noisy channel model [2] is a framework that can be applied in plenty of scenarios such as machine translation and spell checkers. These models are composed of two components: a source model and a channel model. E. Brill et al. described [4] an improvement to noisy channel spelling correction via a more powerful model which works by learning generic string to string edits, along with the probabilities of each of these edits. This more powerful model has a much better performance in accuracy over previous approaches to noisy channel spelling correction.

In this paper, we build a ranking model which not only takes account of the string to string edits and probabilities, but also introduces the language model features generated from corpus and GMM model features that reflect the touch behaviors. After feature engineering, we trained a LambdaMART model [3] to assign score to the candidate words given by the input method. These words are then re-ranked in accordance with the scores, which makes the most expected word to rank top above all the candidate words displayed to users in the input method. Considering factors from various aspects comprehensively, the final model makes a prominent improvement on the accuracy of the input method and ameliorates the input experience.

2 Methods

The model framework is shown in Figure 2, which can be divided into three parts: input to word prediction module, feature engineering and LambdaMART model. The input to word prediction module mainly consists of a base language model and a noisy channel model. The language model provides a batch of potential candidates prefixed with the input character sequence and assigns a numerical value to each candidate due to the probability distribution learned from corpus. The noisy channel model aims to find the intended words given an input where the letters have been scrambled in some manner. After gaining all the possible candidates, certain selected features are calculated and the LambdaMART model is used to rank the candidate words according to the features.

2.1 Feature Engineering

Feature engineering revealing the relativity of the input string and candidate words is fundamental to the application of LambdaMART. Combining practical results and the domain knowledge of the data, three types of features are utilized to represent the underlying structure of each candidate.

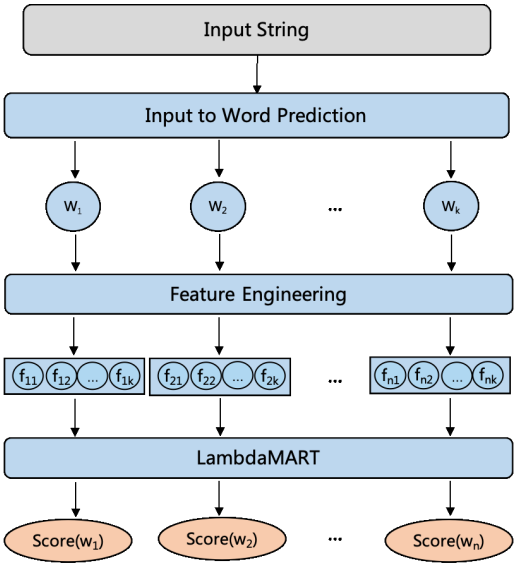

Figure 2: The IME ranking framework

2.1.1 Language Model

- N-gram probability:

  An n-gram model is a type of probabilistic language model for predicting the next item in such a sequence in the form of a Markov model [9]. A trigram model is trained here to approximate the probability of a word given all the previous words $P(w_n|w_{n-1}\cdots)$ by using only the conditional probability of the two preceding words $P(w_n|w_{n-1}w_{n-2})$ due to Markov assumption.

- Transition probabilities based on POS:

  A part of speech is a category of words that have similar grammatical properties[8]. Depending on a large part of speech tagged corpus, we trained a probabilistic language model for approximating the probability of a POS given the POS of the two preceding words. The conditional probabilities that reveal the underlying syntactic relevance between context and candidate words avail the following LambdaMART ranking.

- Character-level perplexities:

  In information theory, perplexity is a measurement of how well a probability distribution or probability model predicts a sample. Here a similar perplexity model is made use of which is based upon character level rather than word level. Given a certain candidate $C = c_1c_2\cdots c_n$, the character-level perplexity that measures the reasonableness of this word can be estimated by:

$$ppl_{char} = \exp\frac{-\log p(c_1c_2\cdots c_n)}{n} \quad (1)$$

  The character-level perplexities fairly reflect the rationality of candidate words and comparatively benefit the performance of the LambdaMART.

2.1.2 Error Model

There may be candidate words generated from error correction. For this type of words, error model estimates the probabilities of error correction.

In [4], an improvement is proposed by learning generic string to string edits, along with the probabilities of each of these edits. In keeping with the idea of this model, we select a set S consisting of s_i, w_i string pairs that have relativity high frequency in corpus, representing spelling errors s_i paired with the correct spelling of the word w_i. For each $\{s_i, w_i\}$ string pairs, we calculated the transition probabilities $P(s \to w)$.

After gaining the string pairs S and corresponding probabilities from corpus, input string along with the candidate words which can be find in the pre-trained pairs set S will be assigned with the feature representing the probabilities of edits from input string to candidate words.

2.1.3 GMM: Layout Model

Using mobile devices equipped with touch screen, incorrect touches occur frequently in the input progress due to the limited size of screen. For instance, we find it common that the adjacent keys of the target character are apt to be touched by mistake and most of these incorrect touches locate on the border of the objective button. Under the circumstances, modeling these specific touch positions properly makes for improving the precision of correction.

For each input action, a sequence of touch operations that may contain substitution error can be obtained. To take advantage of these touch information, we trained GMM (Gaussian Mixture Model) [7] which can map the input touch sequence to the character sequence according to the touch positions. Utilizing this pre-trained GMM model, the probabilities of candidate words given the corresponding input touch sequence can be calculated. These probabilities that reveal the pertinence of input behavior and candidate words are the third type of features for the ranking model. The process of modeling and feature generating will be discussed in details below.

For each input action, a series of touch point positions can be collected. Generally, a touch point can be represented as $v = (x, y)$, where (x, y) is the normalized coordinate according to the screen resolution to eliminate the difference of various de-

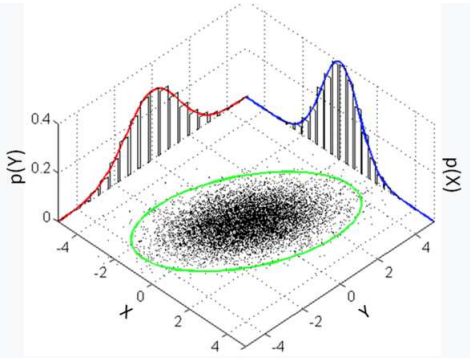

Figure 3: The Gaussian model of touch position on keyboard button

vices. Given a touch point vector v, the probability of button b_v is denoted as $p(b_v|v)$. Figure 3 shows the touch point distribution of a button from the volunteers' feedback. According to the figure, we can make an assumption that the touch point coordinates of each button follow a two-dimensional Gaussian distribution. Therefore, for a given v, the probability that the corresponding key is c among K keys can be described in GMM format:

$$p(c|v) = \frac{\phi N(v|\mu_c, \theta_c)}{\sum_{j=1}^{K} \phi N(v|\mu_j, \theta_j)} \quad (2)$$

Given the input touch sequences $E{=}v_1 v_2 \cdots v_m$, each candidate word is a sequence of characters $C{=}c_1 c_2 \cdots c_m$. Assuming that each type activity is independent, we can calculate $p(C)$ for each candidate:

$$p(C) = \prod_{i=1}^{m} p(c_i|v_i) \quad (3)$$

This probability reflecting the latent correlation between the expected word and the touch points provides the LambdaMART model with special information about the input behaviors itself.

2.2 Re-ranking

Aiming at sorting the candidate words optimally, we utilize the LambdaRank algorithm [3] to present the top one candidate that satisfies the user's expectation most. LambdaMART is the boosted tree version of LambdaRank, which is based on RankNet. RankNet, LambdaRank, and LambdaMART have proven to be very successful algorithms for solving ranking problems [6]. The algorithm combines the gradient boosting tree and the following loss function:

$$C = \sum_{\{i,j\} \in I} |\Delta NDCG_{ij}| \log(1 + e)^{-\delta(s_i - s_j)} \quad (4)$$

For the sake of better performance of LambdaMART, we take detailed and well-chosen information into account and feed them together to the LambdaRank model to learn an appropriate ranking for the input-candidate pairs. The valuable information includes language model features, error-correction features and GMM model features. Acquiring the underlying patterns of these information, LambdaMART assigns a specific score for each input-candidate pair, which is regarded as the basis of ordering. The top one word above the ranking result for the k candidate texts will be considered as the most possible one the user would like to type.

In order to construct the training dataset, input behaviors are collected from volunteers' input log, which mainly consist of three parts: the expected word of per input behavior, the sequence of input characters and the corresponding coordinates of each touch point. For each entry of input behavior, k candidate words can be generated from the language model and the noisy channel model. Associating each candidate word with the corresponding input behaviors, features can be extracted for k candidates and then k pieces of training data are produced. It is worth mentioning that in the k candidates only the one matching the expected word gets a score of 1 while the rest get a score of 0. Hence for each input-candidate pair $(input, candidate)$, features x can be generated by uniting the input with the candidate as described before, and the corresponding score $y = \{1, 0\}$. After data cleaning and feature extraction, 800,000 touch sequences are used to train the LambdaMART model and 100,000 touch sequences are separated for testing for each language.

3 Experimental Result

After training models for two languages respectively, the LambdaMART models are used to assign proper scores to re-rank candidates in test dataset which contains 100,000 touch sequences for each language. The ranking results are then evaluated by NDCG and $F_{0.5}$ (which weights precision twice as much as recall) [10, 11]. NDCG here measures the ranking quality and $F_{0.5}$ is used as a comprehensive indicator to fully evaluate the effect of correction.

As Table 1 and Table 2 show, the LambdaMART model distinctly enhances the performance of the input method comparing with the LM + EM method, which is the base line on the strength of language model and error model approaches. For all two languages, there has been a significant promotion on $F_{0.5}$ and NDCG is improved to a certain degree as well.

	English	Russian
LM+EM	0.8907	0.8659
LM+LambdaMART	0.9049	0.8749

Table 1: The NDCG performance on English and Russian datasets

	English	Russian
LM+EM	0.6631	0.6439
LM+LambdaMART	0.7360	0.7515

Table 2: The $F_{0.5}$ performance on English and Russian datasets

	English	Russian
Third-Party	0.8794	0.8361
LM+EM	0.8854	0.8442
LM+LambdaMART	0.8913	0.8555

Table 3: The NDCG performance on English and Russian datasets without keystroke

	English	Russian
Third-Party	0.6608	0.6969
LM+EM	0.6544	0.6451
LM+LambdaMART	0.7365	0.7112

Table 4: The $F_{0.5}$ performance on English and Russian datasets without keystroke

To compare with the third-party input method, similar experiments are carried out on third-party input method for English and Russian. Because of the inconsistency of the keyboard size and layout in two input methods, the information about touch points may not be of use upon the third-party input method. For the sake of fairness, GMM features are not calculated and utilized in the feature engineering module in the contrast experiment. As shown in Table 3 and Table 4, the performance of the LambdaMART model without information of key strokes surpasses the third-party input method both in NDCG and $F_{0.5}$.

4 Conclusion

In this paper, we implement a ranking model based on LambdaMART for improving the input method accuracy on mobile devices. This ranking model combines selected features from various aspects and re-ranks the candidate words to display the most expected word. Deployed in Baidu's input tool on mobile services, this solution has been proven effective and practical in several languages such as English and Russian. In future work, we are going to expand this approach in other languages like Japanese.

References

[1] Hootsuite & We Are Social (2019). Digital 2019 Global Digital Overview. retrieved from https://datareportal.com/reports/digital-2019-global-digital-overview.

[2] Shannon, Claude Elwood. "A mathematical theory of communication." Bell system technical journal 27.3 (1948): 379-423.

[3] Burges C, From RankNet to LambdaRank to LambdaMART: An overview[J]. Learning, 2010, 11(23-581): 81.

[4] Brill E, Moore R C. An improved error model for noisy channel spelling correction, ACL, 2000: 286-293.

[5] Zhang J, Wang X, Feng Y, et al. Fastinput: Improving input efficiency on mobile devices, CIKM, 2018: 2057-2065.

[6] Chapelle O, Chang Y. Yahoo! learning to rank challenge overview[C]//Proceedings of the learning to rank challenge. 2011: 1-24.

[7] Biernacki C, Celeux G, Govaert G. Assessing a mixture model for clustering with the integrated completed likelihood[J]. IEEE transactions on pattern analysis and machine intelligence, 2000, 22(7): 719-725.

[8] Toutanova K, Klein D, Manning C D, et al. Feature-rich part-of-speech tagging with a cyclic dependency network, NAACL, 2003: 173-180.

[9] Chen S F, Goodman J. An empirical study of smoothing techniques for language modeling[J]. Computer Speech & Language, 1999, 13(4): 359-394.

[10] Jarvelin K, Kekalainen J. Cumulated gain-based evaluation of IR techniques[J]. ACM Transactions on Information Systems (TOIS), 2002, 20(4): 422-446.

[11] Dahlmeier D, Ng H T. Better evaluation for grammatical error correction, NAACL, 2012: 568-572.