The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Search Action Sequence Modeling with Long Short-Term Memory for Search Task Success Evaluation

Alin Fan,[1] Ling Chen,[1, 2, *] Gencai Chen[1]

[1] College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China
[2] Alibaba-Zhejiang University Joint Institute of Frontier Technologies, Hangzhou 310027, China
{fanalin, lingchen, chengc}@cs.zju.edu.cn

Abstract

Search task success rate is a crucial metric based on the search experience of users to measure the performance of search systems. Modeling search action sequence would help to capture the latent search patterns of users in successful and unsuccessful search tasks. Existing approaches use aggregated features to describe the user behavior in search action sequences, which depend on heuristic hand-crafted feature design and ignore a lot of information inherent in the user behavior. In this paper, we employ Long Short-Term Memory (LSTM) that performs end-to-end fine-tuning during the training to learn search action sequence representation for search task success evaluation. Concretely, we normalize the search action sequences by introducing a dummy idle action, which guarantees that the time intervals between contiguous actions are fixed. Simultaneously, we propose a novel data augmentation strategy to increase the pattern variations on search action sequence data to improve the generalization ability of LSTM. We evaluate the proposed approach on open datasets with two different definitions of search task success. The experimental results show that the proposed approach achieves significant performance improvement compared with several excellent search task success evaluation approaches.

Introduction

Search task success rate is one of the prime metrics based on the search experience of users to measure the performance of search systems, which has received considerable attention in recent years (Ageev et al. 2011; Odijk et al. 2015). A search task is an atomic information need, which results in one or more queries (Hassan 2012). From the aspect of search log data used, existing studies working on search task success evaluation could be roughly classified into two categories, i.e., modeling search task success with pure activity logs (e.g., issue a query and click a document) (Hassan, Jones, and Klinkner 2010; Hassan 2012) and modeling search task success with activity and text logs (e.g., the content of queries and web pages) (Ageev et al. 2011; Wang et al. 2014; Jiang et al. 2015). Compared with the latter approaches, the former approaches that model search task success via using pure activity logs could preserve the privacy of users at a higher level. In this paper, we focus on search task success evaluation by using pure activity logs.

A search task could be represented as an action sequence, which consists of search actions and times between two contiguous actions (Hassan 2012). Modeling search action sequence would help to capture the latent search patterns of users in successful and unsuccessful search tasks (Hassan, Jones, and Klinkner 2010). Existing researches working on modeling search action sequence with pure activity logs for search task success evaluation could be roughly classified into two categories, i.e., static model and Markov chain models (Hassan, Jones, and Klinkner 2010; Hassan 2012). Static model uses aggregated features to describe the user behavior and leverages general classifiers in evaluation. However, static model depends on heuristic hand-crafted feature design and ignores a lot of information inherent in the user behavior. Markov chain models provide a more accurate picture of the user behavior by modeling the transitions between search actions, which strongly relies on the conditional independence assumption (i.e., in a search action sequence, the probability of moving to the next action only depends on the present action and not on the previous actions). However, the conditional independence assumption cannot be satisfied in all search tasks. In a search process, the user learns and filters information constantly. She would consider all the information that has been collected to determine the next search action. Therefore, the factors that affect the next search action of users should not only take into account the current search action, but also consider the previous search actions. All these weaknesses motivate us to find a more complete representation of search action sequences to effectively characterize the multifarious search behavior patterns of users.

* Corresponding author

In this paper, we employ Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) that performs end-to-end fine-tuning during the training to learn search action sequence representation for search task success evaluation. Though LSTM has shown the remarkable ability recently in modeling sequential data in image captioning (Chen et al. 2017), sentiment classification (Yang et al. 2017), action recognition (Zhu et al. 2016), etc., it has not been exploited in the field of search task success evaluation. This might be attributed to two challenges: First, search action sequence is a temporal event (each action could be considered as an event) sequence (Luo et al. 2015), and the lengths of dwell times between contiguous actions are not equal. Since LSTM considers by default that the time intervals between contiguous events are fixed in the sequential data, performing LSTM directly on raw datasets of search action sequences would lose the characteristics of time. Second, the amount of human-labeling search tasks is limited due to the expensive and time consuming collection process, which is more inclined to make the model over-fitting.

To cope with the first challenge, we assign each action a fixed length of dwell time t_{idle}, and then introduce a dummy idle action with time t_{idle} to split the residual dwell times into small pieces. Therefore, the time intervals between contiguous actions are fixed and the characteristics of time are incorporated into the sequential data as well.

To address the second challenge, we propose the Dwell Time Perturbation (DTP), a novel form of data augmentation on search action sequence data. Data augmentation is a crucial way to reduce over-fitting by artificially enlarging datasets with label-preserving transformations, which has been widely used in neural network based pattern recognition tasks. Typically for computer vision (Cireşan et al. 2011) and image recognition (Krizhevsky, Sutskever, and Hinton 2012), the transformations (e.g., translating, rescaling, and distorting) have led to significant improvement in recognition performance. Different from the above strategies, DTP introduces a fluctuation factor t_{flu} to slightly disturb the dwell times and then generates new search action sequences.

Our contributions can be summarized as follows:

- Introduce a dummy idle action to represent dwell times in search action sequences, which guarantees that the time intervals between contiguous actions are fixed;
- Propose a novel data augmentation strategy DTP to increase the pattern variations on search action sequence data to improve the generalization ability of LSTM;
- Employ LSTM to learn search action sequence representation for search task success evaluation and achieve significant performance improvement compared with several excellent approaches on open datasets.

Related Work

In this section, we review related work in two major areas, i.e., search task success evaluation and data augmentation.

Search Task Success Evaluation

Search task success is defined as the fulfillment of an atomic information need during interactions with search engines (Field, Allan, and Jones 2010). Some work in this area focuses on designing features and employs general classifiers in evaluation. Ageev et al. (2011) exploited a Conditional Random Field (CRF) model with a set of additional querying and browsing features to evaluation search task success. Feild et al. (2010) combined search interaction features as well as information from other sensors to predict frustration. Wang et al. (2014) established a latent structural learning framework with well-designed structured features to model action-level satisfaction for search-task satisfaction prediction. Guo et al. (2012) tried to capture the patterns of fine-grained interactions (e.g., mouse cursor movements and scrolling) to help evaluating search satisfaction. However, features used in these pervious researches are all designed by human knowledge, which need the effort of domain experts and might miss some useful information that has not been developed yet. In addition, well-designed features rely on more search information recorded in search logs, which would touch the deeper privacy of users.

Other work focuses on modeling search action sequence by using pure activity logs. Hassan et al. (2010) utilized Markov chain to model search action sequence and showed that modeling action transitions is better than using simple statistical features to evaluate search task success. Later, Hassan (2012) improved the Markov chain model by taking the prior distribution into consideration. However, these models strongly rely on the conditional independence assumption, which cannot be met in all search tasks. In this paper, we concerned on modeling search action sequence by taking full advantage of search action and dwell time data.

Data Augmentation

Data augmentation is essential to teach the network the desired invariance and robustness properties, when only few training samples are available. Data augmentation using label-preserving transformations has been widely used in neural network based tasks in computer vision, image recognition, speech recognition, etc. In computer vision and image recognition, the label-preserving transformations are easy to conceive, e.g., translation, deformation, and reflection (Krizhevsky, Sutskever, and Hinton 2012; Chatfield et al. 2014). In the area of speech recognition, Jaitly et al. (2013) proposed VTLP that generates addition-

al samples via perturbing the spectral data. Later, VTLP with both random (Ragni et al. 2014) and deterministic (Tüske et al. 2014) perturbation has been used in different systems and both have improved performance.

Although the methodology of data augmentation has not been developed in search task success evaluation, inspired by the work of Jaitly and Hinton (2013), we consider disturbing the dwell times of search action sequences slightly to obtain additional training data.

Problem Definition

The aim of search task success evaluation is to estimate whether the search task ended up being successful or not. In the rest of this section, we give the definitions of terms that are used throughout this paper.

Definition 1. *Action:* An action a is an interaction that a user performs with a search engine, e.g., committing a query string to a search engine and clicking an interested document in the returned search engine result page.

Definition 2. *Dwell time:* A dwell time, denoted by t, is a time interval between each two contiguous search actions.

Definition 3. *Search action sequence:* A search action sequence S of a search task is an ordered sequence of search actions along with the dwell time between these actions. Given a search task with n actions, the search action sequence could be expressed as $S = < a_1, t_1, a_2, t_2, \ldots, a_{n-1}, t_{n-1}, a_n >$. Specifically, a_n is a pre-defined dummy action that indicates the end of a search task.

Definition 4. *Search task success:* Given a search task, search task success is a binary label y. If the information need of the user has been met, the search task is successful and $y = 1$. Otherwise, the search task is unsuccessful and $y = 0$.

Various approaches have been proposed to identify the boundaries of search tasks from search logs (Boldi et al. 2008; Wang et al. 2013; Li et al. 2016). In this work, we consider the search task identification has been performed in a pre-processing step.

Methodology

In this section, we show the details of the proposed approach.

Overall Framework

The framework of the proposed approach is illustrated in Figure 1. White circles are search actions. Since the available labeled search tasks are sparse, we use the DTP based data augmentation strategy to obtain more labeled search tasks. After that, a dummy idle action (denoted as I) is introduced to represent dwell times in search action sequences. The black circle represents the idle action I. Then, the time intervals between contiguous actions in search action sequences are fixed. Since the length of each search action sequence is ragged, we pad the sequences with 0 (the grey circles) in the front of each sequence. Afterwards, the search action sequence representation could be produced along with the pre-trained action embeddings that generate the representation of each search action. In the end, the LSTM network is employed for sequence classification. The output layer is a dense layer with one mode, and the activation function is sigmoid.

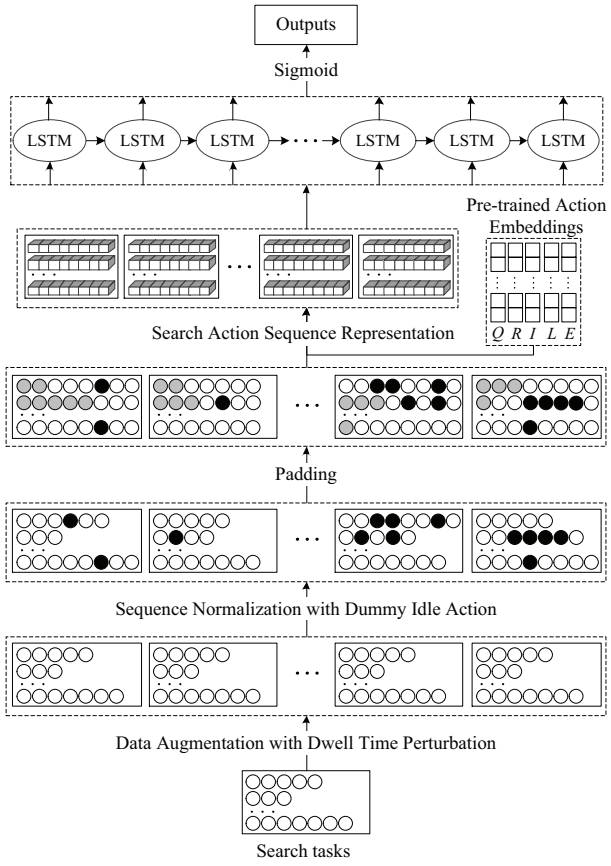


Figure 1. The Framework of the Proposed Approach.

Data Augmentation with Dwell Time Perturbation

Data augmentation based on label-preserving transformations can help to alleviate the sparse data issue. Label-preserving transformations artificially generate more training samples by transforming the existing training samples using certain forms of transformations that preserve the class labels. The aim of this paradigm is to increase the pattern variations through the transformations to improve the classification invariance and the generalization ability of the neural networks. In spired by the work of Jaitly and Hinton (2013) that used a warping factor randomly chosen at a fixed range to warp the frequency axis, we consider to

introduce a fluctuation factor t_{flu} to slightly disturb the dwell times and then generate new search action sequences. The choice of t_{flu} is based on the following hypothesis:

Hypothesis 1. Given a fluctuation factor $t_{flu} \in (0, 1)$ and a search action sequence $S = < a_1, t_1, a_2, t_2, \ldots, a_{n-1}, t_{n-1}, a_n >$, we have $S' = < a_1, (1 \pm t_{flu})t_1, a_2, (1 \pm t_{flu})t_2, \ldots, a_{n-1}, (1 \pm t_{flu})t_{n-1}, a_n >$. If $y(S) = label$, then $y(S') = label$, where y is the success label of a search action sequence and $label \in \{0, 1\}$.

The idea behind this hypothesis is that transforming a raw search action sequence in the way of slightly disturb the dwell times would not change the label. Imagine that when a user issues a query to a search engine, and stays in the search engine result page for 2 seconds, i.e., $t_1 = 2$, if $t_{flu} = 0.2$, then $t_1 = t_1'$, $t_1' \in \{1.8, 2.2\}$. Practically, users could not feel significant difference among staying 2, 1.8, and 2.2 seconds.

Data augmentation could be applied at training processes, test processes, or both (Chatfield et al. 2014). We try to apply DTP at both training and test processes. At the test process, for each test sample, the multiple probabilistic predictions of the variants generated from the same test sample were combined by averaging the class probabilities[1], and the final classification result of a search task S is computed as:

$$y_S = \left\lfloor \frac{1}{2} + \frac{1}{m+1}(P^S + \sum_{i=1}^{m} P_i^S) \right\rfloor \quad (1)$$

where m is the number of variants generated from S via DTP, P^S is the class probability of S, and P_i^S is the class probability of the ith variant.

Sequence Normalization with Dummy Idle Action

Since the dwell times in a search action sequence are not equal in general, we need to normalize the raw search action sequence data to guarantee that the sequence could retain the characteristics of time as much as possible. Specifically, a fixed length of dwell time t_{idle} is assigned to each search action in a sequence beforehand. After that, a dummy idle action I with time t_{idle} is introduced to replace the residual dwell times with a certain amount of action I. Given a search action sequence with n actions, the number of action I between each two contiguous actions is:

$$N_{idle} = \lceil t_i / t_{idle} \rceil - 1, \ i \in (1, n-1) \quad (2)$$

According to the alphabet in Table 1, we show an example of search action sequence normalization in Table 2. In the original search action sequence, the user issues a query to the search engine, and then she examines the search engine result page for 5 seconds. After that, she clicks a result and stays on that page for 14 seconds. Afterwards, she clicks a result again and stays for 2 seconds. Finally, she ends her search. The pure-action version of the original sequence does not take dwell times into consideration. The normalized version of the original sequence replaces the dwell times with a certain amount of action I according to Formula 2.

Action	Description
Q	submit a query to a search engine
R	click on a returned document in search engine result page
I	stay in the current action for a fix time (a dummy action proposed in this paper)
L	click on a hyperlink in the current clicked document (not in a search result page)
E	end the search task

Table 1. An Alphabet of Action Types Used to Encode Search Actions.

Sequence Form	Detail
Original sequence	$Q\ {}_{5s}\ R\ {}_{14s}\ R\ {}_{2s}\ E$
Original sequence without time	$Q\ R\ R\ E$
Sequence normalization	$Q\ I\ R\ I\ I\ I\ I\ R\ E$

Table 2. An Example of Search Action Sequence Normalization ($t_{idle} = 3$).

Search Action Sequence Representation

Word embeddings (Mikolov et al. 2013; Pennington, Socher, and Manning 2014) are a family of natural language processing techniques designed to map semantic meaning to a geometric space, i.e., each word in a dictionary could be represented by a numeric vector; to a certain extent, the distance between any two vectors would characterize the semantic relationship between the two words. In this paper, we consider each action (Q, R, I, L, E) as a word, and use the entire labeled search action sequences as the training text. We employ word2vec (Mikolov et al. 2013) to obtain the semantic relationship of each action pair.

Overview of LSTM

The LSTM model is proposed to solve the problem of vanishing gradient in conventional recurrent neural networks (Hochreiter and Schmidhuber 1997; Hu et al. 2017). A LSTM is a sequence of units that share the same set of parameters. There are three sigmoid gates that control the reading, writing, and memory updating: input gate i_t, forget gates f_t, and output gate o_t, respectively. Each LSTM unit has a memory cell c_t. The memory is updated through reading a new input x_t and the previous output h_{t-1}. Then an output states h_t is written based on c_t. The gates and states are computed as follows:

[1] We also tried to combine the predictions by geometrically averaging the class probabilities. The experimental results show that the gap between the two combination strategies is extremely small.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (3)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (4)$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad (5)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (6)$$

$$h_t = o_t \odot \tanh(c_t) \quad (7)$$

where σ denotes sigmoid function, $\odot$ denotes element-wise product, W_* is the transformation matrix from the input to LSTM states, U_* is the recurrent transformation matrix between the recurrent states h_t, and b_* is the bias vector.

In this paper, we treat LSTM as a black box technique that receives search action sequence data as input sequential data.

Experiments and Results

In this section, we present the experiments to empirically evaluate the proposed approach.

Data

We perform experiments using the dataset described in the work of Ageev et al. (2011). They designed a game-like online contest for crowdsourcing search behavior studies[2]. In their study, 156 participants were asked to perform several predefined informational tasks via a search game interface, and submit the answers they found to the system. Search behaviors of all the participants were logged (including the player ID, query string, timestamp, and the corresponding clicks) and annotated by the authors according to different success criteria. The data were collected in four different game rounds. According to our search task success definition, we treat the following two kinds of tasks as success:

• *Answer correct*: the user believed she found (and then submitted) the answer and the answer was correct;

• *Answer sent*: the user believed she found (and then submitted) the answer and the answer could be correct or not.

Basic statistics of these datasets are given in Table 3.

Success Definition	# $\text{Task}_{\text{successful}}$	# $\text{Task}_{\text{unsuccessful}}$
Answer correct	971	516
Answer sent	1294	193

Table 3. Basic Statistics of Evaluation Datasets.

Baselines

We compare the proposed approach with three best-performing approaches. The first baseline (Field, Allan, and Jones 2010) poses the problem as a classic machine learning problem and uses a logistic regression model to predict search frustration. We built a logistic regression model based on the search action and time features adopted from previous researches (Ageev et al. 2011; Hassan 2012; Jiang et al. 2015). The features we used in this paper are listed in Table 4. We refer to this approach as "LogistRegress".

Descriptions of Features
Number of query-related search actions
Number of click-related search actions
Number of algorithmic results clicks
Number of hyperlink clicks
Numbers of transitions between each two consecutive actions[3]
Time span of a search task
Average dwell time
Average time to the first click-related search action
Average time between clicks-related search actions
Average number of query-related search actions per second
Average number of click-related search actions per second

Table 4. The Features of Search Action Sequences for Search Task Success Evaluation.

The second baseline (Hassan 2012) trains two Markov Chain models to describe the search patterns of users in successful and unsuccessful search tasks, respectively. To concatenate action transitions and dwell time, Hassan (2012) replaced every query and click action with one of two different actions based on the dwell time in the current state of action. We use the same setting in our experiments and denote this baseline as "ImpMML".

The last baseline (Ageev et al. 2011) employs CRF to evaluate search task success based on a set of behavior features. Strictly speaking, it is not fair to compare the proposed approach with CRF. Because CRF not only use the search action sequence data, but also introduce some text information, e.g., query features (query word length, stop words, etc.), engine name (google, bing, and yahoo), and search engine result page features (click position, domain name, etc.). Since Hassan (2012) found the performance of CRF and the ImpMML depends on the success definition, i.e., for some definitions the ImpMML performs better and for others CRF wins better performance, we would like to explore the performance difference between CRF and the proposed approach.

Experimental Setup

We consider the successful and unsuccessful classes as the positive and negative classes, respectively. We use the proposed DTP to augment each dataset by 10 times, and

[2] http://ir-ub.mathcs.emory.edu/uFindIt/

[3] For example, given a search task "$Q\ R\ E$", we can extract the following two transitions: $Q \rightarrow R$ and $R \rightarrow E$. Generally, we get $(|A|-1)^2$ features where A is the set of possible actions. According to the experimental datasets, we have $A = \{Q, R, L, E\}$.

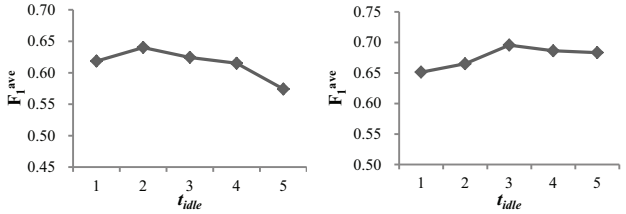
then we have 14870 (1487 × 10) search tasks. We take accuracy and F_1 score as the evaluation metrics. Concretely, we use macro-averaged (Manning et al. 2009) accuracy, F_1^{suc} (F_1 score for the successful class), F_1^{unsuc}(F_1 score for the unsuccessful class), and F_1^{ave} (the arithmetic mean of F_1^{suc} and F_1^{unsuc}) as the evaluation metrics to measure the performance of each approach. We utilize Leave-One-Subject-Group-Out (LOSGO) cross-validation (Schuller et al. 2010) employed in the work of Ageev et al. (2011) (they denoted LOSGO cross-validation as a conditional *k*-fold cross-validation) to evaluate the results. LOSGO cross-validation can guarantee subject independent in the evaluation process (Steidl et al. 2005). In detail, for each of the four game rounds, we use search tasks from the other three game rounds as the training data (We use 25% of the training data as a validation set), and use the current game round as the test data. Therefore, we get four groups. For each group, neither the users nor the predefined informational tasks intersect. In each group, we further random split the test data to 8 subsets with equal size, and use the same model learned from the corresponding training data to test these subsets. Finally, for each approach, we have 32 test results (8 results per group × 4 groups). We use 2-tailed paired t-test to compute the statistical significance of any observed differences.

We set window = 3, size = 128, and sg = 1 for action embeddings. For the configuration of LSTM on each dataset, we use sigmoid for activation function, binary_crossentropy for loss function, adam with batch size of 128 for optimization, and dropout rate 0.35 for hidden layers. During the 1000 epochs, we use early stopping when the loss of the validation set starts to increase with patience = 10. We report results for 128 embedding dimensions and 16 hidden units. We fix the seed on each dataset for reproduction.

Influence of Parameter Setting

In the proposed approach, finding an appropriate fixed length of dwell time t_{idle} is vital. Empirically, we vary the value t_{idle} from 1 to 5. To avoid introducing additional noise, we limit the fluctuation of dwell time to a small range (t_{flu} = 0.1).

Figure 2 shows the performance of the proposed approach (we denote the proposed approach as DTP+LSTM) with different t_{idle} on two datasets. We find that the worst results always appear when t_{idle} = 1or t_{idle} = 5. On the one hand, as t_{idle} decreases, the amount of actions increases and the lengths of sequences extend. It makes the representation of search action sequences become complicated and might ask more labeled data to help learning. On the other hand, as t_{idle} increases, the benefits of time characteristics would diminish. When t_{idle} is large enough, the dummy idle action would be removed from search action sequences, and DTP degenerates to replicate the raw sequence data. Eventually, the best t_{idle} for *Answer correct* is 2, and for *Answer sent* the best t_{idle} is 3, which are fixed in the following experiments.

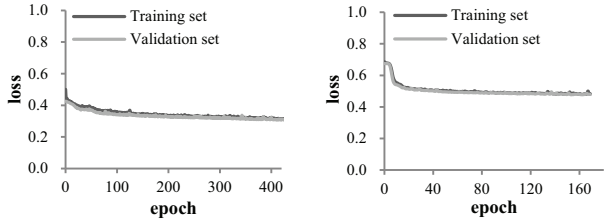


(a) Answer correct (b) Answer sent

Figure 2. The Performance of DTP+LSTM with Different t_{idle}.

Convergence Process

Figure 3 shows the convergence processes of the proposed approach on two datasets. Since we employ the strategy of early stopping to avoid over-fitting, for each dataset, the proposed approach terminates learning on a different epoch. On the two datasets, the losses on the training set and the validation set are extremely close to each other, and the losses decrease as the epoch grows until the stopping criterion has been met, which shows the stability and effectiveness of the proposed approach.



(a) Answer correct (b) Answer sent

Figure 3. The Losses of DTP+LSTM on the Training Set and the Validation Set.

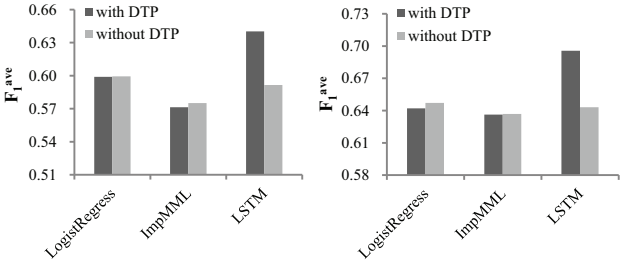
Results

The Impact of Performing DTP.

Figure 4 shows the performance of LogistRegress, ImpMML, LSTM, and their DTP versions for search task success evaluation on two datasets. We notice that by performing the DTP process on raw search action sequence data, the performance of LSTM for evaluating search task success has been greatly improved on all datasets, which implies that the proposed data augmentation program could significantly reduce the generalization errors of LSTM.

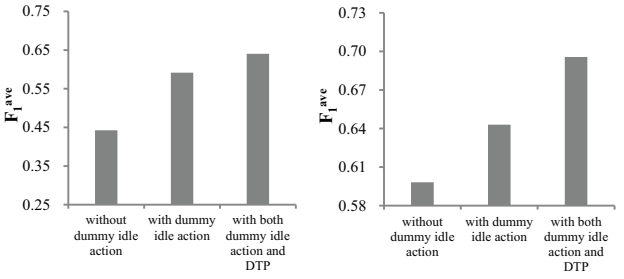
Since the CRF (Ageev et al. 2011) employs additional text information, the DTP strategy could not apply on these data. We just investigate the impacts of DTP on LogistRegress and ImpMML. The astonishing phenomenon is that on the datasets of *Answer correct* and *Answer Sent*, we

cannot see any performance improvement. On the contrary, the performance of the two baselines tends to slightly decline. That might be because the additional sequence data generated from DTP process contain slight noise of dwell time, and the two baselines do not have enough learning ability on the two datasets to discover additional representation from the corresponding augmentative data.



(a) Answer correct (b) Answer sent

Figure 4. The Performance of LogistRegress, ImpMML, LSTM, and Their DTP Versions for Search Task Success Evaluation.



(a) Answer correct (b) Answer sent

Figure 5. The Performance of LSTM without Dummy Idle Action, LSTM with Dummy Idle Action, and LSTM with both Dummy Idle Action and DTP.

The Benefit of Introducing the Dummy Idle Action.

The dummy idle action is used to the transform dwell times into actions in search action sequences. Since the amount of original search action sequences is small compared with that of the augmented datasets, the convergence speed of LSTM is relatively slow. Therefore, we use early stopping with patience = 50. Figure 5 shows the performance of LSTM without dummy idle action, LSTM with dummy idle action, and LSTM with both dummy idle action and DTP (DTP cannot perform without introducing the dummy idle action) on two datasets. We find that LSTM has the worst performance on the sequence data without introducing dummy idle action. The reason is that without the dummy idle action, the search action sequence data would degenerate to the sequences with raw search actions, and directly performing LSTM on these data would lose the features of dwell time. In addition, the dummy idle action is crucial for augmenting datasets. After a slight disturbance to dwell times, the dummy idle action with an appropriate fixed length t_{idle} would help generating new search action sequences and thus enhance the generalization ability of LSTM. Therefore, LSTM wins the best performance via the setting of both dummy idle action and DTP for search task success evaluation.

Comparison with the Baselines.

When comparing the performance of different machine learning algorithms, it should be ensured that these algorithms perform on the same dataset (with or without data augmentation) to guarantee the differences in performance of these algorithms are not due to the gaps of data volume. We have observed that the proposed strategy of data augmentation cannot provide additional benefits to the baselines on certain datasets. Therefore, for LogistRegress and ImpMML, we report the better results that are obtained from each dataset with/without DTP.

Tables 5 and 6 show the performance (F_1^{suc}, F_1^{unsuc}, and F_1^{ave}) of DTP+LSTM and the baselines on two datasets, respectively. * indicates whether DTP+LSTM is statistically superior to the compared approach (for a significant level of $\alpha = 0.05$). We find that the proposed DTP+LSTM exhibits significant performance improvement over other baselines on all datasets, which implies LSTM could learn more plentiful representations of search pattern by using search actions and dwell times. The performance of LogistRegress, ImpMML, and CRF depends on the success definition, and the difference in performance is small in all datasets.

Approach	Accuracy	F_1^{suc}	F_1^{unsuc}	F_1^{ave}
LogistRegress	0.684	0.780	0.419*	0.599*
ImpMML	0.676*	0.776	0.374*	0.575*
CRF	0.674*	0.768*	0.433*	0.601*
DTP+LSTM	**0.703**	**0.787**	**0.493**	**0.640**

Table 5. The Accuracy, F_1^{suc}, F_1^{unsuc}, and F_1^{ave} of the Proposed Approach and the Baselines on the Answer Correct Dataset.

Approach	Accuracy	F_1^{suc}	F_1^{unsuc}	F_1^{ave}
LogistRegress	0.878	**0.932**	0.362*	0.647*
ImpMML	0.870	0.926	0.347*	0.637*
CRF	0.864*	0.923*	0.326*	0.625*
DTP+LSTM	**0.881**	**0.932**	**0.459**	**0.696**

Table 6. The Accuracy, F_1^{suc}, F_1^{unsuc}, and F_1^{ave} of the Proposed Approach and the Baselines on the Answer Sent Dataset.

Conclusions and Future Work

To find a more effective representation of search task to modeling search task success, we propose a search task success evaluation approach based on LSTM. We embed the dwell time representations in search action sequences via introducing a dummy idle action, and propose a novel

strategy of data augmentation to make the best use of the limited search action sequence data. This paper is the first attempt to solve the search task success evaluation problem with deep neural networks. Our findings suggest that deep neural networks in general should be considered in the future for this problem and other similar problems. The proposed approach evaluates search task success by exploiting pure activity log data, which implies it is particularly applicable to the areas where privacy requirements are more stringent, e.g., evaluation in email search that only activity log data are allowed to use (Kim et al. 2017). In the future work, we will investigate how to incorporate fine-grained behaviors (e.g. mouse hovering and scrolling) into the proposed approach.

Acknowledgments

This work was supported in part by China Knowledge Centre for Engineering Sciences and Technology (No. CKCEST-2014-1-5), the Natural Science Foundation of China (No. 61332017), and the Science and Technology Department of Zhejiang Province (No. 2015C33002).

References

Ageev, M.; Guo, Q.; Lagun, D.; and Agichtein, E. 2011. Find It If You Can: A Game for Modeling Different Types of Web Search Success Using Interaction Data. In *SIGIR*, 345-354.

Boldi, P.; Bonchi, F.; Castillo, C.; Donato, D.; Gionis, A.; and Vigna, S. 2008. The Query-flow Graph: Model and Applications. In *CIKM*, 609-618.

Chatfield, K.; Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2014. Return of the Devil in the Details: Delving Deep into Convolutional Nets. *arXiv preprint arXiv: 1405.3531*.

Chen, M.; Ding, G.; Zhao, S.; Chen, H.; Liu, Q.; and Han, J. 2017. Reference Based LSTM for Image Captioning. In *AAAI*, 3981-3987.

Cireşan, D. C.; Meier, U.; Masci, J.; Gambardella, L. M.; and Schmidhuber, J. 2011. High-performance Neural Networks for Visual Object Classification. *arXiv preprint arXiv: 1102.0183*.

Field, H.; Allan, J.; and Jones, R. 2010. Predicting Searcher Frustration. In *SIGIR*, 34-41.

Guo, Q.; Lagun, D.; and Agichtein, E. 2012. Predicting Web Search Success with Fine-grained Interaction Data. In *CIKM*, 2050-2054.

Hassan, A. 2012. A Semi-supervised Approach to Modeling Web Search Satisfaction. In *SIGIR*, 275-284.

Hassan, A.; Jones R.; and Klinkner, K. L. 2010. Beyond DCG: User Behavior as a Predictor of a Successful Search. In *WSDM*, 221-230.

Hochreiter, S., and Schmidhuber, J. 1997. Long Short-term Memory. *Neural Computation* 9(8): 1735-1780.

Hu, L.; Li, J.; Nie, L.; Li, X. L.; and Shao, C. 2017. What Happens Next? Future Subevent Prediction Using Contextual Hierarchical LSTM. In *AAAI*, 3450-3456.

Jaitly, N., and Hinton, G. E. 2013. Vocal Tract Length Perturbation (VTLP) Improves Speech Recognition. In *ICML Workshop on Deep Learning for Audio, Speech and Language*, 625-660.

Jiang, J.; Hassan Awadallah, A.; Shi, X.; and White, R. W. 2015. Understanding and Predicting Graded Search Satisfaction. In *WSDM*, 57-66.

Kim, J. Y.; Craswell, N.; Dumais, S.; Radlinski, F.; and Liu, F. 2017. Understanding and Modeling Success in Email Search. In *SIGIR*, 265-274.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. Imagenet Classification with Deep Convolutional Neural Networks. 2012. In *NIPS*, 1097-1105.

Li, L.; Deng, H.; He, Y.; Dong, A.; Chang, Y.; and Zha, H. 2016. Behavior Driven Topic Transition for Search Task Identification. In *WWW*, 555-565.

Luo, D.; Xu, H.; Zhen, Y.; Ning, X.; Zha, H.; Yang, X.; and Zhang, W. 2015. Multi-Task Multi-Dimensional Hawkes Processes for Modeling Event Sequences. In *IJCAI*, 3685-3691.

Manning, C. D.; Raghavan, P.; and Schütze, H. 2009. *Introduction to Information Retrieval*. Cambridge, England: Cambridge University Press [Online]. Available: https://nlp.stanford.edu/IR-book/

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *NIPS*, 3111-3119.

Odijk, D.; White, R. W.; Awadallah, H. A.; and Dumais, S. T. 2015. Struggling and Success in Web Search. In *CIKM*, 1551-1560.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*, 1532-1543.

Ragni, A.; Knill, K. M.; Rath, S. P.; and Gales, M. J. 2014. Data Augmentation for Low Resource Languages. In *Interspeech*, 810-814.

Schuller, B.; Vlasenko, B.; Eyben, F.; Wollmer, M.; Stuhlsatz, A.; Wendemuth, A.; and Rigoll, G. 2010. Cross-corpus Acoustic Emotion Recognition: Variances and Strategies. *IEEE Transactions on Affective Computing* 1(2): 119-131.

Steidl, S.; Levit, M.; Batliner, A.; Noth, E.; and Niemann, H. 2005. "Of All Things the Measure is Man" Automatic Classification of Emotions and Inter-labeler Consistency [Speech-based Emotion Recognition]. *Acoustics, Speech, and Signal Processing*, I-317-I-320.

Tüske, Z.; Golik, P.; Nolden, D.; Schlüter, R.; and Ney, H. 2014. Data Augmentation, Feature Combination, and Multilingual Neural Networks to Improve ASR and KWS Performance for Low-resource Languages. In *Interspeech*, 1420-1424.

Wang, H.; Song, Y.; Chang, M. W.; He, X.; Hassan, A.; and White, R. W. 2014. Modeling Action-level Satisfaction for Search Task Satisfaction Prediction. In *SIGIR*, 123-132.

Wang, H.; Song, Y.; Chang, M. W.; He, X.; White, R. W.; and Chu, W. 2013. Learning to Extract Cross-session Search Tasks. In *WWW*, 1353-1364.

Yang, M.; Tu, W.; Wang, J.; Xu, F.; and Chen, X. 2017. Attention Based LSTM for Target Dependent Sentiment Classification. In *AAAI*, 5013-5014.

Zhu, W.; Lan, C.; Xing, J.; Zeng, W.; Li, Y.; Shen, L.; and Xie, X. 2016. Co-Occurrence Feature Learning for Skeleton Based Action Recognition Using Regularized Deep LSTM Networks. In *AAAI*, 3697-3703.