DeepProbe: Information Directed Sequence Understanding and Chatbot Design via Recurrent Neural Networks

Zi Yin*
Stanford University
350 Serra Mall
Stanford, CA 94305
zyin@stanford.edu

Keng-hao Chang
Microsoft
1020 Enterprise way
Sunnyvale, CA 94084
kenchan@microsoft.com

Ruofei Zhang
Microsoft
1020 Enterprise way
Sunnyvale, CA 94084
bzhang@microsoft.com

ABSTRACT

Information extraction and user intention identification is a central topic in modern query understanding and recommendation systems. In this paper, we propose DeepProbe, a generic information-directed interaction framework which is built around an attention-based sequence to sequence (seq2seq) recurrent neural network. DeepProbe can rephrase, evaluate, and even actively ask questions, leveraging the generative ability and likelihood estimation made possible by seq2seq models. DeepProbe makes decisions based on a derived uncertainty (entropy) measure conditioned on user inputs, possibly with multiple rounds of interactions. Three applications, namely a rewritter, a relevance scorer and a chatbot for ad recommendation, were built around DeepProbe, with the first two serving as precursory building blocks for the third. We first use the seq2seq model in DeepProbe to rewrite a user query into one of standard query form, which is submitted to an ordinary recommendation system. Secondly, we evaluate DeepProbe's seq2seq model-based relevance scoring. Finally, we build a chatbot prototype capable of making active user interactions, which can ask questions that maximize information gain, allowing for a more efficient user intention idenfication process. We evaluate first two applications by 1) comparing with baselines by BLEU and AUC, and 2) human judge evaluation. Both demonstrate significant improvements compared with current state-of-the-art systems, proving their values as useful tools on their own, and at the same time laying a good foundation for the ongoing chatbot application.

KEYWORDS

Deep Learning; RNN; Seq2Seq; ChatBot; Recommendation system; Attention Mechanism; Online Advertising; Sponsored Search; Query Rewriting; Probabilistic Scoring; Information Gain

1 INTRODUCTION

Recent years have witnessed a boom in deep learning, which revolutionizes areas including computer vision, speech recognition and natural language processing. One widely-used deep learning model is the sequence to sequence (seq2seq) model, which demonstrated their power in machine translation [16], achieving higher BLEU score than conventional methods like phrased-based statistical machine translation models.

Different bells and whistles have been developed which further boost the performance of seq2seq models, one prominent example of which is the attention mechanism. In [9], the authors proposed this mechanism, which augments the top hidden vector at the decoder side with a weighted average of the encoder hidden vectors. The weights can be calculated through a cosine similarity or a generalized matrix inner product, where the weight matrix is part of the parameters to be learnt. By adding attention to the deep seq2seq model, the authors were able to better align inputs and outputs, and subsequently achieve an additional 5.0 improvement in BLEU score.

On top of natural languages, seq2seq models can be trained with literally any kind of paired sequence data. Authors in [18] build a model with IT helpdesk question-answer conversation log so that it will read new user questions and respond "machine-translated" answers. Authors in [7] take email correspondence log to build a model that suggests email reply candidates for users to choose from in mobile environment. Reflecting that a question-answering system would need a knowledge base to search answers from, we propose a staged approach that can leverage existing recommend system as is, serving as the knowledge base to search for the right answer. We apply the seq2seq model to understand user questions, using it to rewrite the question to one in standard query form that an ordinary recommendation system would understand. The rewrite is submitted to the recommendation system to retrieve a set of candidate answers. We show that with attention mechanism the model can rewrite questions with better quality measured by BLEU score. We also show that those rewrites can retrieve ads from a commercial search engine with better human labeled quality, proving that the system has significant commercial values.

Another powerful aspect of the seq2seq model besides its generativeness, namely its statistical property as likelihood estimators, have not been fully investigated by previous work. We built a seq2seq likelihood estimator in DeepProbe, which serves as the central model for an information directed evaluation and interaction framework. When used as a evaluation tool, a posterior probability derived from the seq2seq likelihood estimator will be calculated which serves as a relevance criterion. We can use it to refine candidates returned by a recommendation system. By comparing it against existing baselines like CDSSM [12], we find significant performance improvement evaluated by AUC on a manually labeled dataset. When used as a interactive tool, the seq2seq

*This work was completed during the first author's internship at Microsoft.

KDD'17, August 13–17, 2017, Halifax, NS, Canada.

DOI: http://dx.doi.org/10.1145/3097983.3098148

model steers an agent through the user interaction process. An agent like a chatbot, tries to identify the intent of a user at a interactive session. The agent, using the seq2seq estimator, calculates the conditional entropy through a Naive Bayes procedure, which will be updated every time new information comes, i.e. a new user input. The agent iteratively uses the information to make a decision to either make a recommendation or ask further questions to gather more information. We build a chatbot prototype using this framework. The prototype is built on a commercial search engine which recommends product ads. The chatbot will recommend an ad if a user asks questions with product intent. When the user intention is not clear, it actively asks the user by formulating questions around product attributes that maximize the expected information gain.

The contribution of the paper is summarized as follows. We introduce DeepProbe, an information-directed interaction framework built upon a seq2seq model. We propose and implement a practical way to answer user questions in a staged approach: (1) we apply seq2seq model to understand and rewrite user questions into one that an ordinary recommendation system can understand and return candidates, (2) we use seq2seq model to score and pick better candidates, and finally (3) we use seq2seq to derive confidence measure and probe users for clarification if necessary.

2 MODELS

2.1 Deep Multi-layer Seq2Seq Attention Model

We use a seq2seq neural network enhanced with attention mechanism, which is illustrated in Figure 1. A seq2seq model is comprised of an encoder and a decoder, each consisting of several vertically stacked layers. Below we give a detailed explanation.

2.1.1 Embedding Layer. The embedding layer takes a word and converts it to its vector representation. The parameter required for this layer is a matrix $W_{emb} \in \mathbb{R}^{d_{emb} \times |\mathcal{V}|}$. Specifically, when a word with index i is given to the embedding layer, it produces $W_{\cdot,i}$, the i-th column of the matrix, which is a dimension d_{emb} vector. We learn separate embedding layers and parameters for the encoder and decoder, i.e. two W_{emb} matrices.

2.1.2 Variable-depth LSTM Recurrent Layers. The LSTM recurrent layer with depth l consists of l vertically stacked LSTM blocks. Each LSTM block takes three inputs: e_t, c_{t-1} and h_{t-1}, where e_t is the input from below, c_{t-1} and h_{t-1} are inputs from the previous step. Its output, h_t, is computed in the following way:

$$\begin{aligned}
i_t &= \sigma(W_{ei} e_t + W_{hi} h_{t-1} + b_i) \\
f_t &= \sigma(W_{ef} e_t + W_{hf} h_{t-1} + b_f) \\
c_t &= f_t \cdot c_{t-1} + i_t \cdot \tanh(W_{ec} e_t + W_{hc} h_{t-1} + b_c) \\
o_t &= \sigma(W_{eo} e_t + W_{ho} h_{t-1+} b_o) \\
h_t &= o_t \cdot \tanh(c_t)
\end{aligned}$$

where $\cdot$ denotes the element-wise product between vectors. LSTM is an enhanced recurrent neural network (RNN) that addresses short-term memory issue of a vanilla RNN, by maintaining additional cell vector c_t and introducing input gate i_t, forget gate f_t, and output gate o_t. Detailed discussions of the advantages of LSTM can be found in [6], which is omitted in this paper due to the space limit. For the lowest LSTM layer, e_t is the output of embedding layer with dimension d_{emb}, so $W_{e*} \in R^{d_h \times d_{emb}}$, $W_{h*} \in R^{d_h \times d_h}$, and $b_* \in R^{d_h}$ are the parameters to be learned. For the upper LSTM layers, $W_{e*}, W_{h*} \in R^{d_h \times d_h}$, and $b_* \in R^{d_h}$ are the parameters to be learned. $\sigma(\cdot)$ here denotes sigmoid, a nonlinear activation function. For the encoder, each LSTM block is in fact bi-directional (BLSTM), it outputs concatenated hidden vectors from forward and backward directions, for which the final vector fed to decoder is from the last of both directions in concatenation form: $[\overrightarrow{h}_m; \overleftarrow{h}_{-1}]$. For the decoder, each LSTM block has only forward direction, so readers should interpret d_h accordingly, e.g. d_h in decoder should be twice the size of that in encoder. In encoder each LSTM layer other than the lowest should reduce the input size by half so after concatenation the final output size of each BLSTM layer is the same.

2.1.3 Attention Layer. For every top hidden vector of the decoder, we augment it with an attention vector, g_t, which is obtained by combining the top hidden vectors from the encoder. The attention mechanism will be discussed in section 2.2. After concatenating the attention vector g_t with the output vector of the top LSTM layer h_t, we apply a fully connected layer to reduce the dimension back to the same size as the input hidden vector: $\hat{h}_t = Relu(W_c[g_t; h_t] + b_c)$, where for $Relu$ is the rectified nonlinearity unit, $max(0, \cdot)$. Here the parameters are $W_c \in R^{d_h \times 2d_h}$ and $b_c \in R^{d_h}$. The output $\hat{h}_t$ will be passed to the next layer.

2.1.4 Projection Layer. The projection layer takes the combined hidden and attention vector as input, and outputs a vector of dimension $|\mathcal{V}|$. Its parameters include a weight matrix $W_p \in \mathbb{R}^{|\mathbb{V}| \times d_h}$ and a bias vector $b_p \in \mathbb{R}^{|\mathcal{V}|}$. The output at step t is computed as $v_t = \mathrm{softmax}(W_p \hat{h}_t + b_p)$. Note v is a non-negative vector which sums up to 1, hence it can be viewed as a distribution on the vocabulary $\mathcal{V}$. The likelihood of seeing a specific word with index w_t is the w_t-th element of v_t, which is abbreviated as

$$v_t(w_t) \tag{1}$$

2.1.5 Loss Function. We perform end-to-end training to learn all the aforementioned parameters together. For each pair of a source sequence Src and a target sequence Tgt in the training set, where $Tgt = w_{t_1} ... w_{t_n}$, by first encoding Src through encoder, the loss of this pair is a summation of per-word cross-entropy loss between v_i and the label which is a one-hot indicator vector of each word w_{t_i}.

2.2 Attention Mechanisms

Attention mechanism is a powerful add-on to recurrent neural networks that is intended to combat the long-term dependency issue. Even LSTM and GRU networks, which are designed to have long term dependencies, are prone to missing information that occurred long time ago. The intuition behind attention mechanism is that, at each step of the sequence decoding process, we force the network to look back again at the source sequence to pick up the most relevant hidden vectors, and augment the current hidden vector with this extra piece of information.

To introduce the attention mechanism, we define a few notations for convenience purposes. Let the top hidden vectors for the source sequence be $s_1, \cdots, s_m$, and the top hidden vector of the current target word be h_i. We summarize four variants that we

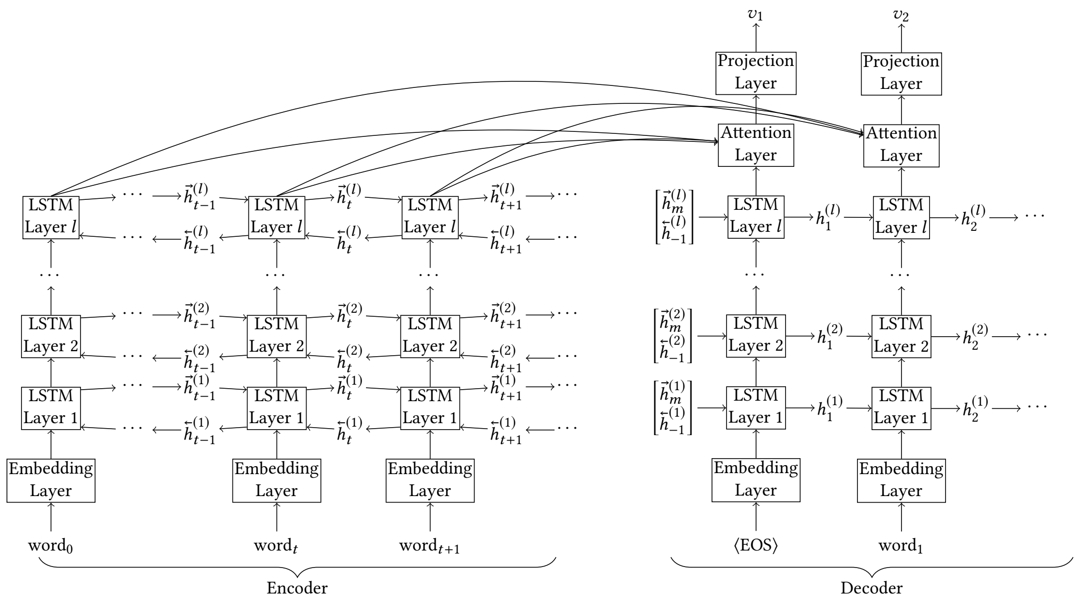

Figure 1: Bi-directional Multilayer LSTM Encoder + LSTM Attention Decoder

intend to experiment [9][14]. These methods are different ways of averaging source hidden vectors $s_1, \cdots, s_m$, where the important words are supposed to have a larger weight, hence it gets the name of "attention". For the four mechanisms, the weight vector $a_{i\cdot}$ for target word i can be calculated respectively firstly:

(1) *(dot)* $\tilde{a}_{ij} = s_j^T h_i$
(2) *(general)* $\tilde{a}_{ij} = s_j{}^T W_g h_i$
(3) *(concat)* $\tilde{a}_{ij} = W_{cc}[s_j; h_i]$
(4) *(tensor)* $\tilde{a}_{ij} = U(s_j{}^T W h_i + V[s_j; h_i] + b)$

Then with $a_{i,\cdot} = \text{softmax}(\tilde{a}_{i\cdot})$, the attention vector is obtained by $g_i = \sum_{j=1}^m a_{ij} s_j$, which will be combined with h_i and fed into the projection layer. Accordingly we learn parameters $W_g \in \mathbb{R}^{d_h \times d_h}$, $W_{cc} \in \mathbb{R}^{1x2d_h}$, tensor $W \in \mathbb{R}^{d_h \times k \times d_h}$ $V \in \mathbb{R}^{k \times 2d_h}$, $b \in \mathbb{R}^k$, and $U \in \mathbb{R}^{1\times k}$.

The four different attention mechanisms aim at different purposes. While *dot* and *general* aim at discovering the similarities between source and target, the last one focuses more on the non-linearity interaction between words, as pointed out in recursive neural network literature [15]. More comparisons and analysis will be discussed in sections 4.

2.3 Likelihood Estimation

The above seq2seq model is capable of giving an estimate of the likelihood of a target sequence, $Tgt = w_{t_1}...w_{t_n}$, given a source sequence Src. First, notice the chain rule for conditional probability, we have

$$Pr(Tgt|Src) = Pr(w_{t_1}|Src) \times \cdots \times Pr(w_{t_n}|w_{t_{n-1}}, .., w_{t_1}, Src)$$

For the i-th word in the target sequence, w_{t_i}, the conditional distribution $Pr(w_{t_i}|w_{t_{i-1}}, .., w_{t_1}, Src)$ is estimated by the seq2seq model as in Equation (1),

$$\widehat{Pr}(w_{t_i}|w_{t_{i-1}}, .., w_{t_1}, Src) = v_i(w_{t_i})$$

Combine the chain rule step with the seq2seq estimator, we obtain the estimated sequence likelihood, which is

$$Pr(Tgt|Src) = \prod_{i=1}^{n} v_i(w_{t_i}) \tag{2}$$

3 INFORMATION-DIRECTED ADAPTIVE SEQUENCE SAMPLING

In the above discussions, we focused on deep learning models and attention mechanisms. However, its ability was mostly investigated in traditional, non-adaptive and one-shot inference scenarios. By non-adaptive and one-shot, we mean that the data are given to the algorithm *as is*, with no control over the data collecting process whatsoever. Most machine learning algorithms are designed to cope with this scenario, but the rise of new interactive channels like chatbot, virtual agents or interactive webpages demand further. An agent, like a chatbot, has to have the adaptivity of talking and raising clarifying questions to a user to reach his or her goal. So our framework is created to address this. By *adaptive*, it means that it is able to dynamically sample the next user input depending on the current estimates, hence will be more directional and less ad-hoc. In other words, it should interpret user intent and knowingly guide the user to achieve the goal in the most efficient way. Next we will explain how DeepProbe integrates the seq2seq model to do the estimation, identify the next sampling direction, and make recommendations when the agent is confident.

3.1 Recommending an Item

Consider a scenario where we would like to make recommendations. Denote π as a prior distribution on the set of all possible items.

In this setting, each $Item$ can be represented as a sequence, for example the title of an ad. Now, suppose k input sequences $Input_1^k$ from a user are revealed, e.g. from k rounds of interactions, the posterior distribution on the set of items should change accordingly, reflecting the fact that more information is provided by the user. Applying the Bayes rule, we have

$$Pr(Item|Input_1^k) = \frac{\pi(Item)Pr(Input_1^k|Item)}{\sum_{Item}\pi(Item)Pr(Input_1^k|Item)}$$

Under a naïve Bayes framework, we assume conditional independence, $Pr(Input_1^k|Item) = \prod_{i=1}^k Pr(Input_i|Item)$. Combining the two expressions, the update rule becomes

$$Pr(Item|Input_1^k) = \frac{\pi(Item)\prod_{i=1}^k Pr(Input_i|Item)}{\sum_{Item}\pi(Item)\prod_{i=1}^k Pr(Input_i|Item)} \quad (3)$$

where we notice that each likelihood term, $Pr(Input_i|Item)$, is given by the seq2seq likelihood estimator in Equation (2).

3.2 Entropy as a Measure of Confidence

3.2.1 Definition and Discussion of Intuition. Entropy is a functional of a probability distribution, which measures how unpredictable the distribution is. We use it to determine the confidence of an agent, or how vaguely the situation is to the agent. It originated from information theory which quantifies the compressibility of a IID random source sequence [3], but has since been widely applied to other fields, including computer vision and speech recognition. For example, the maximum entropy principle, first proposed by Hoch and Skilling [13] [5], has shown extreme success in image reconstruction and de-blurring. The max entropy principle has found applications in speech recognition, where an example is a speech recognition system [11] built by Peters et. al. In NLP, language models, as in [8], are sometimes built around this idea as well. We also point out that in NLP, the notion of *perplexity*, a standard metric used to compare statistical language models and machine translation such as in [16], can be viewed as the exponent of the entropy. Below we give a formal definition of the entropy functional. Notice in the following definition of conditional entropy, it is not averaged across the random variable it conditions on, hence is itself a random variable.

Definition 3.1 (Entropy, Conditional Entropy). Given a pair of discrete random variables (X, Y), where X takes values from a alphabet $\mathcal{X}$ and Y takes value in $\mathcal{Y}$. Denote their joint distribution as $p_{X,Y}(x, y)$ and marginals $p_X(x), p_Y(y)$,

(1) The entropy of X is defined as

$$H(X) = -\sum_{x\in\mathcal{X}} p_X(x)\log p_X(x)$$

(2) The conditional entropy of X given $Y = y$ is

$$H(X|Y = y) = -\sum_{x\in\mathcal{X}} p_{X|Y}(x|y)\log p_{X|Y}(x|y)$$

Finally, we use $H(X|Y) = \sum p_Y(y)H(X|Y = y)$ to denote the *expected* conditional entropy of X given Y.

In general, a large entropy is an indication of the distribution being more widespread. For example, when entropy is maximized, X has a uniform distribution. On the contrary, when entropy is small, the distribution is more concentrated. $H(X) = 0$ effectively means the distribution is deterministic.

3.2.2 Uncertainty of Sequence Posterior Estimation. The conditional entropy can serve as an uncertainty measure of the estimated sequence posterior distribution. Remember in Equation (3), we discussed the posterior update procedure when k user inputs, $Input_1^k$ are observed. We define the *posterior uncertainty* as the entropy of this conditional distribution, $H(Item|Input_1^k)$. A large posterior uncertainty means the estimation is vague, hence more observations are needed before a decision can be made; on the other hand, a posterior uncertainty close to 0 is an indication of the estimation has pretty much converged to its argmax, under which case a sure recommendation is ready to be made. Next we explain how to sample more observations or determine the best question to ask if it's uncertain.

3.3 Information-directed Sampling: Principle of Maximizing Expected Information Gain

3.3.1 Mutual Information. Originated from information theory, the mutual information quantifies how much information can be reliably communicated through a channel. It is a functional on a pair of random variables (X, Y), which is a measure of how much knowledge one can gain of X when Y is revealed. It is defined as the difference between the entropy of X and the conditional entropy of X given Y.

Definition 3.2 (Mutual Information). The mutual information between (X, Y) is

$$I(X; Y) = H(X) - H(X|Y)$$

Similarly, conditioning on a sequence of random variables $Z_1^k = z_1^k$, the mutual information between (X, Y) is

$$I(X; Y|Z_1^i = z_1^i) = H(X|Z_1^i = z_1^i) - H(X|Y, Z_1^i = z_1^i)$$

3.3.2 Information-Directed Sampling Algorithm. Now suppose the agent is able to proactively interact with the user, being able to ask the user with questions and expects answers from the user. To start with, assume there is a set of questions, $\mathcal{Q} = \{Qst_1, \cdots, Qst_q\}$. Following the maximizing information gain principle, we propose Algorithm 1.

We would like to point out that the Algorithm 1 which maximizes the expected information gain at each step, is effectively a greedy uncertainty-reduction algorithm. This observation is stated in the lemma below.

Lemma 3.3. *The information-gain maximizing Qst proposed at step n is also a uncertainty minimizer at step n.*

Proof. Note that

$$I(Qst; Item|Input_1^n) = H(Item|Input_1^n) - H(Item|Qst, Input_1^n),$$

Note $H(Item|Input_1^n)$ does not depend on Qst, as a result, the maximizer of $I(Qst; Item|Input_1^n)$ is immediately a minimizer of $H(Item|Qst, Input_1^n)$ and vice versa. □

A discussion on the application of a Chatbot and question formulation procedure will be discussed in section 4.3.

Algorithm 1 Information-directed Sequence Sampling

for $n = 1, 2, \cdots$ **do**
  The n-th sequence $Input_n$ is collected from the user.
  Estimate the likelihood, $Pr(Input_n|Item)$, using the seq2seq likelihood estimator.
  Update the posterior distribution

$$Pr(Item|Input_1^n) = \frac{\pi(Item)\prod_{i=1}^{n} Pr(Input_i|Item)}{\sum_{Item}\pi(Item)\prod_{i=1}^{k} Pr(Input_i|Item)}$$

  Calculate the conditional entropy $H(Item|Input_1^n)$
  if $H(Item|Input_1^n) < T$ **then**
    Return $\arg\max Pr(Item|Input_1^n)$, the most likely item.
  else
    Choose Qst that maximizes $I(Qst; Item|Input_1^n)$
    Propose Qst to user; wait for user feedback $Input_{n+1}$
  end if
end for

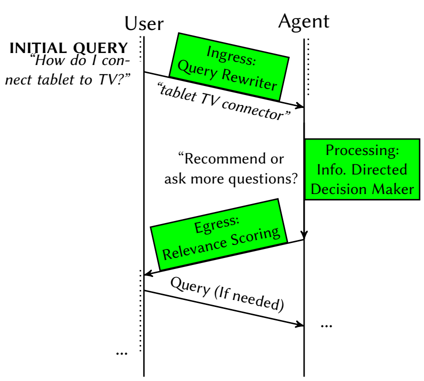

Figure 2: Three applications (in green) of DeepProbe

4 APPLICATIONS

DeepProbe is versatile in the sense that it has a wide range of applicability, covering inference, ranking, adaptive sampling and decision making. We have built three applications of DeepProbe, illustrated in figure 2. In this section, we will elaborate on the details including the design, training, implementation and evaluations of these applications. Note that we build the applications on top of a commercial "product ads" search engine, which recommends products if a user inputs a query with product intent.

4.1 Ingress: Query Rewriting

Query understanding and rewriting is a vital pre-processing step for modern recommendation and information retrieval systems. In the area of product ads recommendation, a user input query in the search engine should be able to 1) trigger an ad recommendation action and 2) return the relevant ads. If the query is in a standard form, like it's grammatically correct, and contains the right keywords, the backend information retrieval system will be able to return the recommendations, and the response time must be short to ensure a high quality of service. In the first application, we apply DeepProbe for query rewriting.

	size	vocabulary	average length	clicks
questions	316K	126K	5.5	782K
queries	481K	870K	2.8	782K

Table 1: Statistics of the Rewrite Training Set

# questions	# queries	# pairs
34K	42K	45K

Table 2: Statistics of the Rewrite Test Set

4.1.1 *What & How: Question Understanding.*
A pain point we identified about product ad recommendation in our search engine is that it does not process queries in question form well. These queries are often ambiguous, and the product is implicitly referred to, usually formulated in a relationship to other entities. As an example, a user might type in the search box a question like

How to connect my tablet to TV?

From a human point of view, this query clearly points to a product: micro HDMI cable. However, this posts a challenge to the information retrieval system, as no clear keywords related to the right product ad were present in the query.

4.1.2 *Training and Data.*
We trained DeepProbe to generate standard queries from question-form queries. To serve this purpose, we used data collected from a "related searches" feature on a commercial search engine. The related searches are a list of queries being recommended to a user when a specific query is typed in the search box, and many of them are standard queries. We picked user-input queries starting with "what" and "how", and regard a related search query as a positve training example if it was clicked by the user. The click behavior by a user confirms that the standard query is indeed relevant to the question the user has entered. By doing so, we were able to collect a dataset consisting of 12 million clicked (question-form query, standard query) pairs. To further focus on questions that will end up with product ad recommendation, we filter the dataset by keeping only the pairs where there was product ad recommendation for the standard query itself. A total of 782 thousand such training pairs were collected. We summarize the statistics of the training dataset in Table 1.

4.1.3 *Details of Model.*
We used the model in section 2.1 with vocabulary size $|\mathcal{V}| = 100k$ for both the encoder and decoder. Any word not in $\mathcal{V}$ is assigned with symbol ⟨UNK⟩. We chose the embedding dimension $d_{emb} = 100$. We used 3-layer LSTMs with hidden vector size d_h=300 on the decoder side, and we implemented 4 different attention scenarios as in Section 2.2. The results for the four different attention mechanisms are compared. The model rewrites to a sequence of words as follows. At step i at the decoder, the model picks the most likely word and use it as the input to the embedding layer at step $i + 1$, until the max length is reached, or an ⟨EOS⟩ token is encountered. We used Theano [17] for model training on a Tesla K20 GPU, with cross entropy as the loss function, and Adadelta [19], a variant of Adagrad [4], for gradient descent. We do end-to-end training to learn all the parameters described in Section 2.1, with a total of 10 epochs.

4.1.4 Result and Evaluation. In our experiments, we found DeepProbe's rewriting helped in two ways. First, while many original queries are product related, they did not trigger product ads, due to the form in which the queries are presented, or their implicitness. After being rewritten, they become more keyword-like and trigger product ads. Some of such examples are

How to connect my tablet to TV → tablet tv connector

How to repair my broken iphone screen → iphone screen replacement

How to charge my iphone → iphone charger

How to protect my iphone screen → iphone screen protector

Secondly, rewriting also helps in retrieving the correct ads, especially when implicit or complex relations are present in the query. To provide an explicit example, the query "How to wire car radio" indicates, from the human understanding perspective, that the user has the radio already and is looking for wiring products. When submitted in the original form, ads on car radios are retrieved. After DeepProbe rewrites it to "radio wiring", the correct ads (radio wiring harness) are retrieved. Another example is the query "How to fix gps in car", where in its original form it triggers ad about mobile GPS, and after rewriting, the correct ads, GPS holders are returned.

As a quantitative evaluation, we test on a different (question-form query, standard query) test dataset. The test set was collected using the same procedure described in Section 4.1.2, but sampled from log in a different time period. In addition, any pair appearing in the training dataset was removed from the test set. We summarize the statistics of the test dataset in Table 2.

4.1.5 Quality of Rewrites. For each pair in the test set, we generate rewrites using different DeepProbe model variations. We evaluate each rewrite against the standard query as baseline using BLEU score [10]. BLEU (bilingual evaluation understudy) is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. We borrow the same technique to evaluate query rewriting since the BLEU score is a standard evaluation for seq2seq model-based translation. We summarize the average BLEU score results in Table 3. We can clearly see that attention mechanisms consistently outperform the base model without attention. This observation is different from several recent attempt in applying seq2seq model for question-answering like [18], which stated that attention mechanism is not helpful. If we reflect based on the examples shown above, this can be explained by the source-target sequence alignment characteristic of our question *rewriting* application, a property that machine translation shares but not question *answering*. Among the attention mechanisms, general attention performs the best while concat attention is the worst. Strictly speaking concat attention does not do alignment directly using the hidden vectors of source and target words. On the other hand, dot and general attention does exactly that. Lastly, the tensor mechanism ranks the 2^{nd}, only a bit worse than general attention. We suspect its structure is too complicated to learn when combined against recurrent neural networks.

4.1.6 Quality of Ad Recommendation. We evaluate the quality of ad recommendation using the rewrites as input to the ad system. We first sampled 1000 question queries from the test pairs, and generate rewrites using different DeepProbe model variations. Then, we submit the rewrites to product ads search engine. The batch submission effort is usually called "scraping" in industry. We also scrape the system with the original question queries and the standard query respectively, in order to compare the ads coverage and quality. The results are presented in Table 4. The ads coverage is defined by % of questions having ads returned. For ads quality, we sample 3000 (original question, ad) pairs for each version of rewrites and their returned ads. Each pair is labeled by a group of trained human judges according to the relevance between the query and the ad. Each label ranges in {bad, fair, good, excellent}, and we consider {fair, good, excellent} as positive. Note that along with the recommended ads, we submit the "original question" to judges. Judges are only comparing the original question to the returned ad, without knowing the ads are actually retrieved using rewrites. The ads quality is based on % positive labeled ads in each 3000-pair set.

Model	BLEU Score
DeepProbe rewrites without attention	0.326
DeepProbe rewrites with dot attention	0.349
DeepProbe rewrites with general attention	**0.388**
DeepProbe rewrites with concat attention	0.331
DeepProbe rewrites with tensor attention	0.364

Table 3: BLEU scores between rewrites and standard queries

In Table 4, we see that only 21.0% of the original questions triggered product ad recommendations. 21.1% of the returned ads are of reasonable quality. If we scrape with the related search queries collected from the log, we see much higher ad coverage at 84.7%. This is expected as we already filter the test set this way. More importantly, we see even better quality ads at 25.3%. This confirms the validity of our rewrite data collection method. The clicked related search queries are indeed relevant to the questions so that, the ads returned using the rewrites are similarly relevant compared to scraping with the questions themselves.

Among DeepProbe's rewrites, we see that using *general* attention achieves both the highest coverage and quality. We see a 3.5x increase in coverage and a 50% increase in quality, relative to the original query. Even when compared with the unobserved ground-truth, i.e., the clicked related search queries, we see only a 10.7% decrease in coverage but a 12.4% increase in quality.

Scrape Set	Ads Coverage	Ads Quality
Original questions	21.0%	21.1%
Related search queries	84.7%	25.3%
DeepProbe rewrites without attention	67.4%	26.7%
DeepProbe rewrites with dot attention	64.2%	33.0%
DeepProbe rewrites with general attention	74.0%	**37.7%**
DeepProbe rewrites with concat attention	64.1%	18.2%
DeepProbe rewrites with tensor attention	64.1%	28.8%

Table 4: Scraping results

4.1.7 Discussion. The results indicate that doing "question"-rewriting with appropriate training data achieves improved recommendation quality and coverage. This staged approach can be

	size	vocabulary	average length	clicks
query	6.4M	68K	4.1	15M
ads	5.1M	114K	9.3	15M

Table 5: Statistics of the Scoring Training Set

# queries	# ads	# pairs	# positive	# negative
23K	915K	965K	234K	731K

Table 6: Statistics of the Scoring Test Set

Implmentation	Decoder	Encoder Architecture	Encoder Embedding	AUC
(a) CDSSM	None	Conv / max pooling	Tri-letter hash	0.726
(b) DeepIntent	None	Conv / max pooling	Word-based	0.728
(c) DeepIntent	None	BLSTM / last pooling	Word-based	0.798
(d) DeepProbe	Yes	BLSTM / last pooling	Word-based	**0.840**

Table 7: AUC scores of different Scoring Frameworks

seamlessly integrated into current infrastructure. It does not require any change in the existing information retrieval system, as the rewritten query can be submitted either instead of or along with the original one. In addition, targeting only "what" and "how" questions is just the first step towards a general-purpose question-answering system. Readers can imagine that this application would be part of a large-scale, comprehensive system, where this application only focuses on product recommendation. Lastly, one may argue that although a significant improvement is observed, the reported ads quality is still not high. This leads to the next section using seq2seq for scoring and keeping better candidates.

4.2 Egress: Relevance Scoring

DeepProbe's ability of estimating items' posterior distribution also makes it a good fit for quality control at the egress side. When a set of ads are returned from the information retrieval infrastructure, DeepProbe can serve as a relevance filter which shows only the most related ads to the user.

4.2.1 Training and Data. The DeepProbe scoring model was trained on our internal dataset, which consists of clicked (query, ad) pairs sampled from a commercial product ad search engine. The ads come from a product ad database, each is a sequence of words describing the corresponding product. The queries are user inputs in our search engine, and if the user clicked on an ad when searching with a query, we regard it as a positive (query, ad) pair. A total of 15 million clicks are sampled from a month long of click logs, which ends up with 6.4 million distinct user queries and 5.1 million distinct ads. We summarize the statistics of the training dataset in Table 5.

4.2.2 Details of Model. We used the model in Section 2.1 and chose a vocabulary size $|\mathcal{V}_q| = 60k$ on the query side and $|\mathcal{V}_d| = 100k$ on the ad side. Any word that is not part of $\mathcal{V}$ is assigned with symbol $\langle$UNK$\rangle$. We chose the embedding dimension $d_e = 150$ on both encoder and decoder sides, and hidden dimension d_h=300 on the decoder side. Although we did notice an improvement in performance for deeper networks, in this experiment we trained a single layer LSTM model for a fair comparison noted below. We trained the model for 5 epochs.

4.2.3 Evaluation. As a quantitative evaluation, we test on a fully annotated test set. The test set contains around 966 thousand (query, ad) pairs where each pair is labeled by a group of trained human judges according to the relevance between the query and the ad. Each label ranges in {bad, fair, good, excellent}. The pairs are sampled from the early selection stage of a commercial ads search engine, where there are a significant amount of low quality selected ads to be pruned out in downstream processing. We use AUC (area-under-curve of the receiver operating characteristic plot) as the metric for evaluation, by considering the good and excellent labels as the positive class and the rest labels as the negative class. This results in a test set consisting of 234 thousand positive and 731 thousand negative pairs. We briefly summarize the statistics of the testset in Table 6. We use a uniform prior π on the set of ads, and hence $Pr(ad|query) \propto Pr(query|ad)$ as in Equation (2). As a result, $Pr(query|ad)$ serves as the relevance score for a pair $(query, ad)$. The AUC is then computed according to the scores for all the pairs in the test set.

We compare against existing relevance scoring baselines, including the popular CDSSM [12] and DeepIntent [20], both of which are deep-learning based and have shown very satisfactory results in production. Given a (query, ad) pair, they encode the query and the ad seperately into two vectors, and then calculate cosine similarity directly from these two vectors as the relevance score. We trained CDSSM, DeepIntent and DeepProbe on the same training dataset, and evaluated the performance by comparing the AUCs on the same testset. Our results are presented in Table 7.

Note both CDSSM and DeepIntent methods only use encoders, unlike in DeepProbe there're both encoder and decoder components. So to make a fair comparison, it becomes necessary to keep the encoder setting as similar as possible, say the encoder architecture, encoded vector size, and depth of recurrent neural networks. The vector size is easy to do and we set it to 300 across the models. We also train all models with depth = 1, and set word-embedding size to 150 if applicable. Below we discuss the different encoder architectures and their AUC performance. We avoid using attention in DeepProbe to keep the comparison fair and simple:

1) DeepIntent with BLSTM resembles DeepProbe the most, i.e. they have the same encoder architecture. They both start with a word-based embedding layer, leverage BLSTM to compute a sequence of hidden vectors, and take the last vector as the final encoded vector. So comparing this against DeepProbe can fairly show the gain by having a decoder. In Table 7, we see DeepProbe achieves 0.84 AUC, as shown in row (d), outperform this baseline with 0.798 AUC shown in row (c).

2) Rather than using BLSTM, at the encoder side, CDSSM uses a convolutional (Conv) layer. The Conv layer aggregates tri-letter-based word-hash vectors via a sliding window. The output is a sequence of vectors which gets further reduced to a final encoded vector with max pooling. In CDSSM's implemenation, it also has a fully connected layer to reduce the size of final encoded vector for online performance reason. To make a fair comparison, we set both the internal hidden vector size and final encoded vector size to 300.

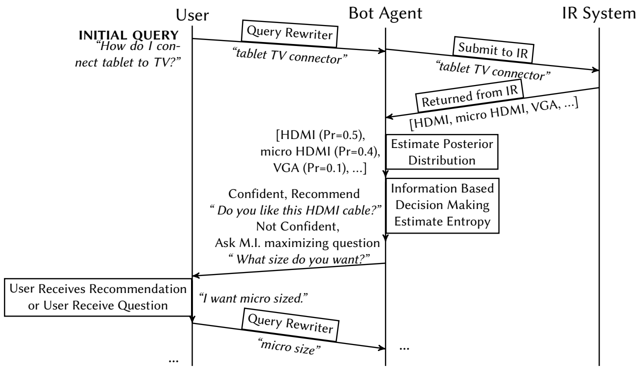

Figure 3: ChatBot Design

In Table 7, we see CDSSM implementation achieves far worse AUC score of 0.726 in row (a).

3) With CDSSM being so different in encoder, namely the tri-letter-based embedding and the Conv layer, we modified DeepIntent implemenation to use Conv layer, in order to understand where the loss in AUC comes from. Specifically, we would like to know whether it is from the different embedding or recurrence layer. After using DeepIntent with Conv layer, the AUC of this implementation achieves only 0.728 AUC, as shown in row (b) of Table 7, similar as CDSSM implementation in row (a). It is strongly suggested, by comparing (b) against (c), that BLSTM-based encoder outperforms Conv-based encoder.

In summary, DeepProbe not only provides scores allowing probabilisitc interpretation, but achieves better performance than similiarity-based scoring methods namely CDSSM and DeepIntent.

4.2.4 Discussion. We would like to briefly discuss the cost of computation and implementation, and point out why DeepProbe is capable of being a good relevance filter. First, from a practical point of view, having DeepProbe acting on top of the existing information retrieval system requires no modification to the infrastructure, minimizing implementation cost. Second, from the computation point of view, when used as a relevance scoring method, DeepProbe needs to calculate a score for each (query, ad_i) pair for all ads requesting a relevance scoring. As a result, the cost of computation grows linearly with the number of ads. This is also the reason we do not use DeepProbe to directly search through the entire ads database for the most relevant ones, as the hundreds of millions of ads in the database can make the search process too long to guarantee service quality. When used as the egress control of a information retrieval engine, however, the number of returned ads are limited; usually at the scale of tens. Moreover, batching and hierarchical softmax can further reduce the computation time required.

4.3 Chatbot: Information Directed Conversation and Recommendation

The last application we introduce is a chatbot that specializes in product ad recommendation. Virtual agents and chatbots have gained popularity due to its user-friendliness and interactiveness. They not only offload some of the jobs of search engines, but also create new user interaction entry points [2]. Below we explain the flow of interaction with examples.

4.3.1 Flow of Dialog and System Behaviors. The interactive session starts with the first query submitted by a user. For example, the user can ask the chatbot "How do I connect my tablet to TV?". The chatbot then retrieves a initial list of related ads from the information retrieval backend system. To do so, it applies DeepProbe's rewriting to the question and convert it into a standard query, in this case "tablet tv connector". This standard query is then submitted to the information retrieval system which returns a list of ads, e.g. ads about HDMI cables, micro HDMI cables, or VGA cables. The chatbot then uses DeepProbe's posterior distribution estimation to calculate the distribution of the returned ad list, and estimates its corresponding conditional entropy. In the decision-making step, if the conditional entropy is less than some threshold T, the top k (3 by default) most relevant ads are returned to the user. Each ad is displayed with a picture, the selling price, the merchant selling it, and embedded with a hyperlink so the user can click on. A user click will redirect the user to the e-commerce web page hosted by the merchant so the user can continue the exploration and make purchase. Otherwise, the bot asks a conditional mutual-information maximizing question to the user, in this case the question is about the "size" of the connector products. For example, "what size do you want?". This conversation goes until a final recommendation is made. Figure 3 gives an illustration of the procedure in the form of a timing diagram. Next we explain how we formulate such questions.

4.3.2 Question Formulation. By the principle of maximizing expected information gain, at each step, if the chatbot is not confident, it is supposed to ask a question that maximized the conditional mutual information. The problem here is, what is the set of questions we are maximizing over for? If we allow arbitrary questions, the chatbot may face issues like 1) the question may be not relevant to the product so is confusing, and 2) the mutual information is difficult to estimate.

To address this issue, we leverage the attributes associated with each ad. For example, an ad about a laptop has attributes "processors", "RAM size", "manufacture" and so on. Similarly for clothes, there are attributes like "color", "size" and "material". By formulating questions based on the attributes, the aforementioned issues go away. Firstly, it will be easier for users to relate. Users will have the perception that the chatbot is working with them to narrow down the most relevant product by confirming the attribute info. Secondly, it is straightforward to estimate mutual information along with attribute-based questions. Notice that attributes only depend on the ads, so $\mathrm{Input}_1^n - \mathrm{Ad} - \mathrm{Attribute}$ forms a Markov chain.

This allows us to estimate mutual information, as the conditional distribution, $Pr(\mathrm{Ad}, \mathrm{Attribute}|\mathrm{Input}_1^n)$, can be calculated by

$$Pr(\mathrm{Ad}, \mathrm{Attribute}|\mathrm{Input}_1^n) = Pr(\mathrm{Attribute}|\mathrm{Ad})Pr(\mathrm{Ad}|\mathrm{Input}_1^n)$$

where the first factor is estimated by counting and the second factor is directly provided by DeepProbe's posterior update. After the information-maximizing attribute is identified, a question will be raised and the user input will be collected to update the posterior distribution again. As an example, if the user is looking for a laptop, a question may look like

What manufacture do you like?

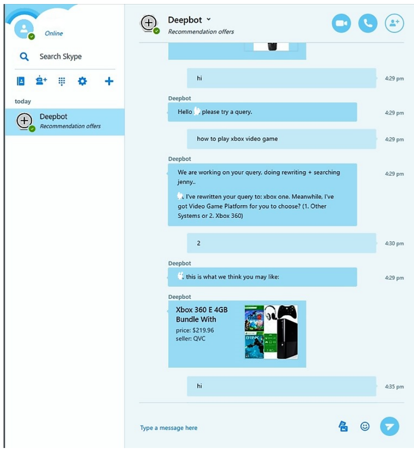

Figure 4: Screenshot of the Chatbot

4.3.3 Implementation and Qualitative Feedbacks. We built the bot using Microsoft Bot Framework [1], which is a chatbot development tool. It supports bot conversation over various platforms, including text messages, Skype, Slack, Messenger, etc. Figure 4 is a screenshot of the chatbot with Skype as the platform.

By demonstrating the prototype to a few colleagues, we got a few encouraging feedbacks. Most of them were surprised by the capability of the chatbot in recommending products when they ask related questions. The "how to connect tablet to tv" case was also a big win. An HDMI ad was recommended back to a user, and by clicking the ad, the title of the redirected web page popped up: "16.4ft Ultra-thin Micro HDMI D to A Long Cable - Connect Tablet / Smart Phone / Mobile / Laptop / Camera to HD TV". Interested users can take this opportunity to learn more about a Micro HDMI cable (it can connect not only tablet but also other devices to TV) and purchase it! Nonetheless, several colleagues pointed out that this chatbot should not be standalone. In addition to recommending products, we should also integrate with other services to provide tutorial videos for example. Last but not least, a colleague asked whether the chatbot can provide information in other verticals other than products. By explaining how the system works, the colleague understood by training on data from a different vertical, and combining with the corresponding search engine, we can generalize the chatbot to where needed.

5 CONCLUSION AND FUTURE WORK

In this paper, we introduced DeepProbe, a sequence-to-sequence model based framework for query understanding, ad recommendation and user interaction. In query rewriting, it significantly increases both the coverage and quality. For relevance scoring, AUC, which is a key metric, surpasses existing systems. It also demonstrated great potential in more efficient user interaction and chatbot design, for which we can rigorously formulate questions to users, based on a principle of maximizing information gain. As an ongoing work, we would like to continue work and experiment on the chatbot, possibly with quantitative experiments for the chatbot. A helpful experiment is that we can measure its efficiency (i.e. number of rounds of interaction) for a user to acquire the information he or she needs.

REFERENCES

[1] 2016. Microsoft Bot Framework Documentation. https://docs.botframework.com/en-us/. (2016). Accessed: 2016-12-25.

[2] Jacob Aron. 2011. How innovative is Apple's new voice assistant, Siri? *New Scientist* 212, 2836 (2011), 24.

[3] Thomas M Cover and Joy A Thomas. 2012. *Elements of information theory.* John Wiley & Sons.

[4] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[5] Jeffrey C Hoch and Alan S Stern. 1996. Maximum entropy reconstruction. *eMagRes* (1996).

[6] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[7] Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufman, Balint Miklos, Greg Corrado, Andrew Tomkins, Laszlo Lukacs, Marina Ganea, Peter Young, and Vivek Ramavajjala. 2016. Smart Reply: Automated Response Suggestion for Email. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD) (2016).* https://arxiv.org/pdf/1606.04870v1.pdf

[8] Sanjeev Khudanpur and Jun Wu. 1999. A maximum entropy language model integrating n-grams and topic dependencies for conversational speech recognition. In *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, Vol. 1. IEEE, 553–556.

[9] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, Lisbon, Portugal, 1412–1421. http://aclweb.org/anthology/D15-1166

[10] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL '02).* Association for Computational Linguistics, Stroudsburg, PA, USA, 311–318. DOI: http://dx.doi.org/10.3115/1073083.1073135

[11] Jochen Peters. 2006. Speech recognition system, training arrangement and method of calculating iteration values for free parameters of a maximum-entropy speech model. (March 7 2006). US Patent 7,010,486.

[12] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web.* ACM, 373–374.

[13] John Skilling and RK Bryan. 1984. Maximum entropy image reconstruction: general algorithm. *Monthly notices of the royal astronomical society* 211, 1 (1984), 111–124.

[14] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 926–934. http://papers.nips.cc/paper/5028-reasoning-with-neural-tensor-networks-for-knowledge-base-completion.pdf

[15] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, Vol. 1631. Citeseer, 1642.

[16] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems.* 3104–3112.

[17] Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688 (May 2016). http://arxiv.org/abs/1605.02688

[18] Oriol Vinyals and Quoc V. Le. 2015. A Neural Conversational Model. *CoRR* abs/1506.05869 (2015). http://arxiv.org/abs/1506.05869

[19] Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* (2012).

[20] Shuangfei Zhai, Keng-hao Chang, Ruofei Zhang, and Zhongfei Mark Zhang. 2016. Deepintent: Learning attentions for online advertising with recurrent neural networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 1295–1304.