Abstract
We consider matching input messages with proper responses for short-text conversation. The matching should be performed not only by the messages and the responses but also by the topics of the messages. To this end, we propose a topic augmented neural network which consists of a sentence embedding layer, a topic embedding layer, and a matching layer. The sentence embedding layer embeds an input message and a response into a vector space, while the topic embedding layer forms a topic vector by a linear combination of the embedding of the topic words whose weights are determined by both themselves and the message vector. The message vector, the response vector, and the topic vector are then fed to the matching layer to calculate the matching score between the message and the response. Empirical study on large scale annotated data shows that our model can significantly outperform state of the art matching models.

1 Introduction
Human-computer conversation is a challenging task in artificial intelligence (AI) and natural language processing (NLP). As early effort towards this goal, Weisenbaum et al. (1966) built a rule-based chatbot in order to pass the Turing test. After that, many chatbots and task-oriented dialogue systems have been designed with more complicated and robust rules (Boden, 2006; Wallace, 2009; Young et al., 2010). Recently, with the large amount of conversation data available on social media, building chatbots with data driven approaches has drawn a lot of attention (Jafarpour et al., 2010; Ritter et al., 2011). In this paper, we study a fundamental problem in building chatbots with data driven approaches, that is how to select a suitable response from candidates for an input message in one round of conversation which is referred to as short-text conversation (STC) in the existing literatures (Ji et al., 2014; Shang et al., 2015).

The success of response selection for STC lies in accurately matching users’ input messages with proper responses. When matching a response with an input message, one needs to consider not only the message and the response but also the topics of the message. This is based on our understanding on how people behave in conversation. When people reply to a message, they may not only follow the literal content of the message, but also associate the message with concepts under the topics of the message. Different concepts have different weights in their mind according to their background knowledge and understanding on the message. With the literal content and the weighted concepts, people generate appropriate responses to the message. Table 1 illustrates our idea with an example. Regarding to “Do you have lunch yet?”, besides saying yes or no, people can also associate their ongoing “diet”, food they like, and restaurants they know with “lunch”, and use them as building blocks of their responses.
to the message. “Diet”, “sandwich”, and “crab-house” are concepts under the topics of the input message, and are more suitable to be talked about in responses than other topically related concepts like “fork” and “knife”.

Existing methods either only rely on the message and the response to perform matching (Wang et al., 2015; Hu et al., 2014), or only use their topics (Lu and Li, 2013), while we aim to mimic people’s behavior in conversation and leverage messages, responses, and topics of the messages for matching. The advantage is that topics can help recognize the relevance between messages and long informative responses like those in Table 1. The challenges include how to obtain the topics of input messages and how to incorporate the topical information into matching. We propose a topic augmented neural network (TANN) for tackling these challenges. TANN is a general framework for incorporating topical information into sentence matching. It consists of a sentence embedding layer, a topic embedding layer, and a matching layer. The sentence embedding layer embeds an input message and a response candidate into a vector space. The topic embedding layer first acquires topic words of the message from a pre-trained Twitter LDA model (Zhao and Jiang, 2011). It then interacts with the sentence embedding layer by leveraging the message vector and the embedding of the topic words to calculate a weight for each topic word. Finally, it forms a topic vector by a weighted average of the embedding of the topic words. The message vector, the response vector, and the topic vector are fed to the matching layer to calculate a matching score between the message and the response. TANN simulates the process in which people pick important topical concepts according to the input message for responding. We implement the sentence embedding layer by a convolutional neural network (CNN) and the matching layer by a multi-layer perceptron (MLP). Thus, our model can enjoy both the powerful matching capability of CNN with MLP and extra topical information provided by a state of the art topic model. Empirical study on large scale annotated data shows that with the extra topical information, TANN can significantly outperform state of the art matching methods for STC.

Our contributions in this paper are three-folds: 1) proposal of leveraging topics as extra information for message-response matching. 2) proposal of a topic augmented neural network for STC in which topics are incorporated into matching by a topic embedding layer. 3) empirical verification of the effectiveness of the proposed method on large scale annotated data.

2 Related Work

Building chatbots for automatically responding to human’s input is always a dream of AI researchers and NLP researchers (Weizenbaum, 1966; Singh et al., 1999; Shaikh et al., 2010). Early work (Weizenbaum, 1966) relied on handcrafted templates for response generation, which requires huge human effort but can only generate limited responses. Recently, researchers begin to develop data driven approaches for building responding machines (Stent and Bangalore, 2014; Ritter et al., 2011). Existing work along this line includes retrieval based methods and generation based methods. The former retrieves response candidates from an index and then employs a matching technique to select the most suitable response for an input message. Several matching algorithms have been proposed (Hu et al., 2014; Wang et al., 2015; Lu and Li, 2013). As representative methods for the latter, Ritter et al. (2011) employed statistical machine translation techniques to generate responses. Shang et al. (2015) formalized response generation as an encoding-decoding process, and learned generation models with recurrent neural networks. Based on these work, context in a conversation has also been taken into account to support multiple rounds of conversation (Lowe et al., 2015; Sordoni et al., 2015). Our work belongs to retrieval based methods. We focus on selecting proper responses for one round of conversation. We propose a neural network based framework which can leverage messages, responses, and topics of the messages for matching.

Convolutional neural networks (CNNs) (Collobert et al., 2011) have been proven effective in many NLP tasks such as text classification (Kim, 2014), entity disambiguation (Sun et al., 2015), answer selection (Yang et al., 2015), tag recommendation (Weston et al., 2014), web search (Shen et al., 2014), sentiment classification (dos Santos and Gatti, 2014), sequence prediction (Li and Liu, 2015) and sentence matching (Hu et al., 2014). In sentence matching, Yin et al. (Yin and Schütze, 2015) proposed a MultiGranCNN to matching a
pair of sentences based on multi granularity information. Pang et al. (Pang et al., 2016) formalized sentence matching as a image recognition task and employed a CNN architecture in the image recognition to predict the matching score. We implement the sentence embedding layer in our model by CNN. Then, our model can leverage the powerful matching capability of CNN for STC which is an important task in sentence matching.

3 Problem Formalization

Before presenting our model, let us first imagine how people react to others’ messages in conversation. When a message comes, people first read the message and understand the message literally. Then they may associate the message with some related concepts under the topics of the message. Some concepts are more proper to be talked about in responses to the message according to their background knowledge and understanding on the message. Finally, they pick important concepts as building blocks and return a response based on both the literal content of the message and the topics of the message. Take the message in Table 1 as an example. To reply to “Do you have lunch yet”, people may first come up with answers like “yes” or “not yet” according to their situations. At the same time, they may think more. They may remember their diet plan; they may want to talk about food they had (e.g., “grilled cheese sandwich”); and they may want to recommend restaurants they know (e.g., “Cutters Crabbouse”). All these concepts are under the same topic with the message. Moreover, compared with diet, food, and restaurants, although concepts like “fork” and “knife” are also related to “lunch”, few people may want to talk about them in their responses to “Do you have lunch yet”? In the creation of replies, people can merge important concepts with “yes” or “not yet” and return responses like those in Table [1]

We aim to simulate how people react to an input message in conversation through a model, and leverage not only the input message but also its topics in matching with responses for short-text conversation. Formally, suppose that we have a data set \( \mathcal{D} = \{(y_i, m_i, r_i)\}_{i=1}^{N} \) where \( m_i \) is an input message, \( r_i \) is a response candidate, and \( y_i \in \{0, 1\} \) is a label. \( y_i = 1 \) means \( r_i \) is a proper response for \( m_i \), otherwise \( y_i = 0 \). Besides \( \mathcal{D} \), we further assume that each \( m_i \) corresponds to \( k \) topics \( T_i = \{t_{i,1}, \ldots, t_{i,k}\} \). \( \forall j, t_{i,j} \) is associated with \( n_i \) concepts. In this work, we restrict the concepts to keywords. Then, \( t_{i,j} \) corresponds to a keyword set \( W_{i,j} = \{w_{i,j,1}, \ldots, w_{i,j,n_i}\} \). Our goal is to learn a matching model \( g(\cdot, \cdot) \) with \( \mathcal{D} \) and \( \{T_i, \cup_{j=1}^{k}W_{i,j}\}_{i=1}^{N} \). For any message-response pair \((m, r)\), \( g(m, r) \) measures their relevance and thus can serve as a criteria in response selection for short-text conversation.

To learn \( g(\cdot, \cdot) \), we need to answer two questions: 1) how to obtain topics and topical keywords of input messages; 2) how to incorporate the topics and the topical keywords into matching. In the following sections, we will first present our method for topic generation, then we will elaborate our matching model and learning approach.

4 Topic Generation

We employ a Twitter LDA model (Zhao et al., 2011) to generate topics and topical keywords for an input message. Twitter LDA belongs to the family of probabilistic topic models (Blei et al., 2003) and represents the state of the art topic model for short texts (Zhao et al., 2011). The advantage of the model is that it can effectively discover latent topics in short texts from social media and group topically related words together. The basic assumption of Twitter LDA is that each short message corresponds to one topic. Each word in the message is either a background word or a topical word under the topic of the message. Specifically, Twitter LDA first draws a multinomial distribution \( \theta \) from a Dirichlet prior \( \text{Dir}(\alpha) \), \( T \) multinomial distributions \( \{\phi^t\}_{t=1}^{T} \) from \( \text{Dir}(\beta) \), a Bernoulli distribution \( \pi \) from \( \text{Dir}(\gamma) \), and another multinomial distribution \( \phi^B \) from \( \text{Dir}(\beta) \). \( \theta \) represents a topic distribution in the entire data set. \( \forall t \in \{1, \ldots, T\} \), \( \phi^t \) is a word distribution under topic \( t \). \( \phi^B \) is a distribution for background words. Given a message \( m \), the model then draws a topic
$z_m$ based on $\theta$. For the $l$-th word $w_{m,l}$ in $m$, an indicator $Y_{m,l}$ is first sampled from $\pi$. If $Y_{m,l} = 1$, then $w_{m,l}$ is a topical word and is sampled from $\phi^w$; otherwise, $w_{m,l}$ is a background word and is sampled from $\phi^B$. Figure 1 gives the graphical model of Twitter LDA.

We estimate $\theta$, $\{\phi^l\}_{l=1}^L$, and $\phi^B$ with a collapsed Gibbs sampling algorithm (Zhao and Jiang, 2011) using $D$ in which we discard irrelevant pairs and concatenate each $m_i$ with relevant $r_i$ to form a short document. After we get the estimations of the parameters, we use them to assign a topic $t_i$ to each $m_i$ in $D$ (i.e., $k = 1$ in Section 3) by the generation process described above. To obtain the keywords related to a topic, we define salience of a word $w$ regarding to a topic $t$ as

$$s(w, t) = \frac{c_w^t}{c_w} c_w^t,$$  

where $c_w^t$ is the number of times that word $w$ is assigned a topic $t$ in the training data and $c_w$ is the number of times that $w$ is determined as a topical word in the training data. Equation (1) means that the salience of a word regarding to a topic is determined by the frequency of the word under the topic (i.e., $c_w^t$) and the probability of the word only belonging to the topic (i.e., $\frac{c_w^t}{c_w}$). $c_w$ acts like IDF in information retrieval and can reduce the importance of common words like “yes” and “cause” to topic $t$. With Equation (1), we select top $n_i$ words ($n_i = 50$ in our experiments) to form the keyword set $W_i$ for message $m_i$ with topic $t_i$.

5 Topic augmented neural network for matching

We propose a topic augmented neural network (TANN) to incorporating topical keywords of input messages into the learning of message-response matching. Figure 2 gives the architecture of our model. TANN is composed of a sentence embedding layer, a topical embedding layer, and a matching layer. A $(m_i, r_i)$ pair in $D$, the sentence embedding layer embeds $m_i$ and $r_i$ into a vector space. Then, the topic embedding layer summarizes the topical keywords in $W_i$ of $m_i$ as a topic vector by a linear combination of the embedding of the keywords. The combination weights are determined by both the embedding of the keywords and the embedding of $m_i$, and are learned from training data. Finally, the message vector, the response vector, and the topic vector are fed to the matching layer to calculate a matching score between $m_i$ and $r_i$. TANN measures the relevance between $m_i$ and $r_i$ by not only the matching of themselves but also the matching of $r_i$ and the topical keywords $W_i$ of $m_i$. It naturally models how people leverage topical concepts in their mind for responding, as we have described in the beginning of Section 3. The framework of TANN is general, as we can implement the sentence embedding layer by any sentence embedding models and the matching layer by any discriminative models. In this work, we choose a convolutional neural network (CNN) (Collobert et al., 2011) as the sentence embedding layer and a two-layer feed-forward neural network (a.k.a multi-layer perceptron (MLP)) as the matching layer. The advantage is that TANN can enjoy both the powerful capability of CNN with MLP on matching a pair of sentences and the extra topical information provided by a state of the art topic model for short texts.

Specifically, for either an input message or a response candidate, CNN looks up an word embedding table and forms a sentence matrix $S = [v_1, v_2, \ldots, v_s]$ as input, where $v_j \in \mathbb{R}^d$ is the embedding of the $j$-th word and $s$ is the maximum length of a sentence. The embedding table is initialized by the results of the skipgram algorithm proposed by Mikolov et al. (2013) and updated during training. Message, response, and topic correspond to different embedding tables. If the length of a sentence does not reach $s$, we put all-zero padding vectors after the last word of the sentence until $s$. CNN separately operates on the message matrix and the response matrix, and represents the input message $m_i$ and the response candidate $r_i$ as a message vector $\tilde{m}_i$ and a response vector $\tilde{r}_i$ respectively by alternating convolution operations and max-pooling operations. Let $z(l,f) = \begin{bmatrix} z_1^{(l,f)}, z_2^{(l,f)}, \ldots, z_s^{(l,f)} \end{bmatrix}$ denote the output of the $l$-th layer under the $f$-th feature map (among $F_l$ of them) where $z_j^{(l,f)} \in \mathbb{R}^{d(l,f)}$. $z_0^{(l,f)} = S$. In convolution, CNN slides a window with width $k^{(l,f)}$ on $z(l,f)$ and splits $z(l,f)$ into several segments. For the $i$-th segment $z_i^{(l,f)} = \begin{bmatrix} z_i^{(l,f)}, \ldots, z_{i+k}^{(l,f)} \end{bmatrix}$, the output of convolution is

$$z_i^{(l+1,f)} = \sigma \left( z_i^{(l,f)} W^{(l,f)} + b^{(l,f)} \right),$$  

where $W^{(l,f)} \in \mathbb{R}^{k(l,f)}$ and $b^{(l,f)} \in \mathbb{R}^{d(l,f)}$ are parameters, and $\sigma(\cdot)$ is an activation function.
In max-pooling, the output of convolution is shrunk in order to enhance robustness. Let $k_j^{l,f}$ denote the width of the window for max-pooling, then the output of max-pooling is

$$z_i^{(l+1,f)} = \max\left( z_i^{(l,f)} \right),$$

where $\max(\cdot)$ is an element-wise operator over vectors. CNN obtains $\vec{m}_i$ and $\vec{r}_i$ by concatenating the vectors in the output of the final layer.

The topic embedding layer begins with a topic word generator, which finds related keyword set $W_i$ of message $m_i$ by what we have described in section 4. After that, we construct a topic matrix $T_i = [\omega_{i,1}, \ldots, \omega_{i,n_i}]^T$ by looking up a word embedding table for each topical keyword in $W_i$. It then calculates weights of the keywords by

$$\omega_i = T_i \cdot A \cdot \vec{m}_i,$$

(4)

where $A$ is a linear transformation learned from training data, and $\forall j, \omega_{i,j} \in \omega_i$ is the weight for the $j$-th keyword in $W_i$. $\vec{m}_i$ is the sentence representation of $m_i$. We further scale $\omega_{i,j}$ to $[0, 1]$ by

$$\alpha_{i,j} = \frac{\exp(\omega_{i,j})}{\sum_{p=1}^{n_i} \exp(\omega_{i,p})}.$$  

(5)

The output of the topic embedding layer is a topic vector $\vec{t}_i$ given by

$$\vec{t}_i = \sum_{j=1}^{n_i} \alpha_{i,j} e_{i,j}.$$  

(6)

From Equation (4), (5), and (6), we can see that the more important a keyword is, the more contribution it will make to the matching of the message and the response. The importance of keywords is determined by both the message and the keywords themselves. The idea here is inspired by the attention mechanism proposed for machine translation (Bahdanau et al., 2014) where important parts of a source language will affect the generation of its translation. We borrow the idea of attention mechanism here, since it can naturally model the interaction between the input message and its topical concepts in responding performed by people.

With $\vec{m}_i$, $\vec{r}_i$, and $\vec{t}_i$, our matching model $g(m_i, r_i)$ is defined by a two-layer feed-forward neural network (i.e., MLP):

$$\sigma_2 \left( w_2 \cdot \sigma_1 \left( w_1 [\vec{m}_i \cdot \vec{r}_i \cdot \vec{t}_i]^T + b_1 \right) + b_2 \right),$$

(7)

where $w_1$, $w_2$, $b_1$, and $b_2$ are parameters. $\sigma_1(\cdot)$ and $\sigma_2(\cdot)$ are activation functions.

We learn $g(\cdot, \cdot)$ by minimizing cross entropy (Levin and Fleisher, 1988) with $D$. Let $\Theta$ represent a set of all parameters in our model, then our objective function $L(D; \Theta)$ is given by

$$-\frac{1}{N} \sum_{i=1}^{N} [y_i \log (g(m_i, r_i)) + (1 - y_i) \log (1 - g(m_i, r_i))],$$

(8)

where $N$ is the number of instances in our training set $D$. We optimize the objective function using back-propagation and the parameters are updated by stochastic gradient descent. As regularization,
we employ early-stopping (Lawrence and Giles, 2000) as it is enough to prevent over-fitting on large scale training data (over 6 million instances). We set the initial training rate and the batch size as 0.001 and 200 respectively.

We implement TANN by an open-source deep learning framework Keras (http://keras.io). We set the dimension of word embedding (i.e., $d$) as 100 and the maximum sentence length (i.e., $s$) as 20. In CNN, we only use one convolution layer and one max-pooling layer, because we find that the performance of the model does not become better with the number of layers increased. The number of feature maps in CNN (i.e., $F$) is 8, and all windows in CNN have a width 3. We use Relu (Dahl et al., 2013) as the activation function $\sigma(\cdot)$ in CNN. In MLP, we set the dimension of the first layer as 800 and the dimension of the second layer as 2 ($g(m, r)$ and $1 - g(m, r)$). Tanh and softmax are chosen as $\sigma_1(\cdot)$ and $\sigma_2(\cdot)$ respectively in MLP.

6 Experiment

6.1 Experiment Setup

We crawled 3,265,912 posts associated with 11,430,692 responses from Weibo. Weibo is a Chinese microblogging service where post-response pairs represent conversation between people. We used the negative sampling technique (Mikolov et al., 2013) to construct the training set $D$. Specifically, given a post $m_i$, we selected one of its associated responses $r_i$ to create a positive instance $(1, m_i, r_i)$. Meanwhile, we randomly sampled another response $r_j$ which is not associated with $m_i$ to create a negative instance $(0, m_i, r_j)$. By this means, we created 6,270,010 training instances with a ratio of positive and negative instances 1:1. Following the same way, we prepared a validation data set with 130,872 instances. There is no overlap between the validation set and the training set.

We compared different models on three data sets. The first one is a negative sampling data set which was constructed in the same way with $D$. The data set contains 130,942 instances and does not have any overlap with $D$ and the validation set. In addition to the negative sampling data, we prepared two human annotated data sets. The first one consists of 422 posts and each post has 30 labeled responses. This data set was published in (Wang et al., 2013). The second one is our in-house data set. We built the data set by the method proposed in (Wang et al., 2013). Specifically, we first indexed the posts with responses in $D$ by an open source Lucene.Net. Then, we crawled another 400 posts that are not contained by $D$ as test posts, and retrieved several similar posts from the index for each test post. Finally, we collected all the responses associated with the similar posts as response candidates for the test post. We recruited three human judges to label if a candidate is proper to be a response of a test post. Each judge labeled a candidate response with 1 if it is a proper one, otherwise the judge labeled it with 0. Each candidate response received three labels and the majority of the labels was taken as the final decision. After removing posts without any proper responses, we finally obtained 328 test posts with 3,418 responses.

On average, each test post has 10.4 labeled responses in the in-house data set.

On the negative sampling data, we employed classification accuracy as a metric. On the human annotated data sets, we employed mean average precision (MAP) (Baeza-Yates et al., 1999), mean reciprocal rank (MRR) (Voorhees and others, 1999), and precision at position 1 (P@1) as evaluation metrics. These metrics are widely used in the existing literatures about text matching. On the negative sampling data, we directly compared different models, while on the human annotated data, we followed Wang et al. (2015) and combined each model with cosine. The combination was conducted by a gradient boosting tree (Friedman, 2001) whose parameters were tuned by five-fold cross validation on the validation data.

6.2 Baseline

We considered the following models as baselines:

Translation Model: We regarded each post in the crawled 3 million data as a source language and its associated response as a target language, then learned word-to-word translation probabilities using GIZA++

Following (Di et al., 2014), we used the translation probability $p(response|tweet)$ to as a matching score between a post and a response.

\[^1\]http://data.noahlab.com.hk/
\[^2\]http://lucenenet.apache.org
\[^3\]The reason why we used another data set for test is that the public data has some bias on posts. For example, there are many posts about “mobile” and “HuaWei”.
\[^4\]http://www.statmt.org/moses/giza/GIZA++.html
**Multi-layer Perceptron**: A post and a response were transformed to vectors by averaging the embedding vectors of the words they contain. The post vector and the response vector were then fed to a two-layer MLP to predict their matching score. The MLP shared the embedding table with our model. The first layer has 100 nodes, and the second layer has 2 nodes.

**DeepMatch_{topic}**: We implemented the matching model in (Lu and Li, 2013) which only used topic information to match a post and a response.

**CNN**: We implemented the CNN model proposed by Hu et al. (2014). The number of feature maps and the width of windows are the same as our model.

**Topic + CNN**: As a naive way to utilize the topical information, we averaged the embedding vectors of the topical keywords in our model as a topic vector, and used it in the input of the matching layer.

All the baseline methods were implemented in the same way as the existing work, and trained using $D$.

### 6.3 Parameter Tuning

There are several parameters we need to determine. For Twitter LDA, following (Zhao and Jiang, 2011), we used $\alpha = 1/T$, $\beta = 0.01$, $\gamma = 0.01$ as the hyperparameters of the dirichlet priors. We tuned the number of topics (i.e. $T$) in $\{20, 50, 100, 200\}$ and the maximum iteration number of Gibbs sampling in $\{100, 200, \ldots, 1000\}$. The best parameter for Twitter LDA is $(200, 1000)$. The number of topical keywords in $W_i$ was tuned in $\{10, 20, \ldots, 100\}$. We chose 50 as the number of topical keywords. We trained the initial word embedding of TANN and the Twitter LDA using $D$, and conducted all tuning on the validation set. The initial word embedding vectors are trained on the $D$.

### 6.4 Quantitative Evaluation

Table 2 and Table 3 report evaluation results on the negative sampling data and the human annotated data respectively. From these tables, we can see that TANN outperforms all baselines in terms of all metrics. The improvements are statistically significantly over all baseline methods (t-test with p value $\leq 0.01$), especially on the human annotated data. On the negative sampling data, since negative instances are easy to identify, all models can achieve a high precision. As a consequence, the performance gap between our model and the baselines is not so big as it on the human annotated data.

| Method          | Score |
|-----------------|-------|
| Random          | 50.00 |
| Translation     | 68.37 |
| DeepMatch_{topic} | 63.42 |
| MLP             | 69.87 |
| CNN             | 75.65 |
| CNN + Topic     | 75.58 |
| TANN            | 76.43 |

**Table 2: Evaluation results on the negative sampling data**

Our model outperforms CNN because it incorporates extra topical information into the matching of messages and responses. The result demonstrates that the topical information is beneficial to the recognition of relevant responses for a message. Our model also performs much better than DeepMatch_{topic}, since it considers both topical similarity and message-response similarity in a vector space and is able to identify those topically similar but irrelevant responses for a message. It is difficult for DeepMatch_{topic} to achieve this as it only considers topical similarity. The result of CNN + Topic is slightly better than CNN but worse than TANN, which demonstrates that topical information is really useful in message-response matching and we provide an effective way to leverage the information. MLP performs badly on these data sets, indicating that simply averaging word embedding cannot model sentences. CNN-based methods are superior to the translation model, because translation model suffers from a sparsity issue. Many term pairs in the test set do not appear in the training set, so the value of generation probability is not accurate on the instances with these pairs.

### 6.5 Qualitative Evaluation

We investigate why TANN can outperform baseline methods on STC with examples in Table 4. The original examples are in Chinese. We translated the Chinese to English here.

Table 4(a) gives an example to show the benefit from topical keywords. The response in the table is relevant to the message "Has anyone watched Christmas in August? Is it worth watching?", since the response is a famous quote in the movie. CNN made a mistake on the relevance between the
message and the response, because it understands the response as talking about love, while the message is talking about a movie. Unlike CNN, TANN first detected topical keywords of the message. Top three keywords are: movie, touching and love, and the weights of the keywords are listed. Since "love" and "touching" have high weights, TANN was able to recognize the relevance between the response and the message.

Table 4(b) gives an example to show the effectiveness of our topic embedding approach. The response in the table is irrelevant to the message, as laughing is not an appropriate word to describe a horror movie. CNN + Topic assigned a high score to the response, because without weights, topical keywords like "awesome" and "movie" reduced the importance of "horror movie" and brought noise to matching. TANN weighed each topical keyword by considering the message. It assigned a high weight to the topical keyword "horror movie". Therefore, it could learn a better topic representation and demoted the response that is not about "horror movie".

Table 4(c) gives an example to show the effectiveness of the function \( s(w, t) \) in the equation 1. The second and the fourth column describe the rank ordered by the frequency of \( w \) given the topic \( t \). Others described the rank ordered by equation 1. We can see general words, such as "what" and "think", are demoted by \( s(w, t) \), which can help to construct a more accurate topic vector.

### Table 3: Evaluation results on the public data and the in-house data

|               | Public data |                | In-house data |                |
|---------------|-------------|----------------|---------------|----------------|
|               | MAP  | MRR  | P@1  | MAP  | MRR  | P@1  |
| Random        | 0.291 | 0.398 | 0.224 | 0.642 | 0.695 | 0.524 |
| Cosine        | 0.514 | 0.690 | 0.554 | 0.597 | 0.662 | 0.553 |
| Translation   | 0.540 | 0.690 | 0.565 | 0.709 | 0.797 | 0.688 |
| DeepMatch\_topic | 0.541 | 0.695 | 0.568 | 0.697 | 0.785 | 0.665 |
| MLP           | 0.541 | 0.702 | 0.564 | 0.694 | 0.786 | 0.665 |
| CNN           | 0.542 | 0.692 | 0.571 | 0.743 | 0.815 | 0.705 |
| CNN + Topic   | 0.548 | 0.706 | 0.564 | 0.744 | 0.818 | 0.708 |
| TANN          | **0.557** | **0.719** | **0.589** | **0.755** | **0.829** | **0.726** |

### Table 4: Qualitative analysis of TANN

(a) Benefit from topical keywords

| Message | Has anyone watched Christmas in August? Is it worth watching? |
|---------|-------------------------------------------------------------|
| Topic words and weight | Movie: 0.499, Love:0.166, touching:0.048 |
| Response | Yep. I love the quote! I always knew that feelings of love would fade just like old photographs. But for you, you will always remain in my heart, as you are in my last moment. |
| CNN Score | 0.283 |
| TANN Score | 0.716 |
| Label | Relevant |

(b) Effectiveness of the topic embedding layer

| Message | I want to watch a FANTASTIC horror movie right now! |
|---------|-----------------------------------------------------|
| Topic words and weight | Horror movie: 0.731, awesome: 0.083, movie:0.032 |
| Response | it is fantastic. I could not stop laughing throughout the whole movie! |
| CNN+Topic Score | 0.865 |
| TANN Score | 0.487 |
| Label | Irrelevant |

(c) Effectiveness of the function \( s(w, t) \)

| Topic 1 | Topic 1’ | Topic 2 | Topic 2’ |
|---------|----------|---------|----------|
| hair    | hair     | dream   | myself   |
| crew cut | now     | sleep   | think    |
| hair style | think   | nightmare | what    |
| hair cut | crew cut | insomnia | night   |
References

[Blei et al.2003] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. The Journal of machine Learning Research, 12:2493–2537.

[Boden2006] Margaret Ann Boden. 2006. Mind as machine: A history of cognitive science. Clarendon Press.

[Collobert et al.2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. The Journal of Machine Learning Research, 12:2493–2537.

[Dahl et al.2013] George E Dahl, Tara N Sainath, and Geoffrey E Hinton. 2013. Improving deep neural networks for lvcsr using rectified linear units and dropout. In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pages 8609–8613. IEEE.

[Levin and Fleisher1988] Esther Levin and Michael Fleisher. 1988. Accelerated learning in layered neural networks. Complex systems, 2:625–640.

[Li and Liu2015] Mingxuan Wang, Zhendong Lu, Hang Li, and Wenbin Jiang. 2015. A convolutional architecture for word sequence prediction.

[Sordoni et al.2015] Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach.
to context-sensitive generation of conversational responses. arXiv preprint arXiv:1506.06714.

[Stent and Bangalore2014] Amanda Stent and Srinivas Bangalore. 2014. Natural Language Generation in Interactive Systems. Cambridge University Press.

[Sun et al.2015] Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI), pages 1333–1339.

[Voorhees and others1999] Ellen M Voorhees et al. 1999. The trec-8 question answering track report. In Trec, volume 99, pages 77–82.

[Wallace2009] Richard S Wallace. 2009. The anatomy of ALICE. Springer.

[Wang et al.2013] Hao Wang, Zhengdong Lu, Hang Li, and Enhong Chen. 2013. A dataset for research on short-text conversations. In EMNLP, pages 935–945.

[Wang et al.2015] Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2015. Syntax-based deep matching of short texts. arXiv preprint arXiv:1503.02427.

[Weizenbaum1966] Joseph Weizenbaum. 1966. Eliza? a computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1):36–45.

[Weston et al.2014] Jason Weston, Sumit Chopra, and Keith Adams. 2014. # tagspace: Semantic embeddings from hashtags.

[Yang et al.2015] Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2013–2018. Citeseer.

[Yin and Schütze2015] Wenpeng Yin and Hinrich Schütze. 2015. Multigrancnn: An architecture for general matching of text chunks on multiple levels of granularity. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL), pages 63–73.

[Young et al.2010] Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. Computer Speech & Language, 24(2):150–174.

[Zhao and Jiang2011] Xin Zhao and Jing Jiang. 2011. An empirical comparison of topics in twitter and traditional media. Singapore Management University School of Information Systems Technical paper series. Retrieved November, 10:2011.

[Zhao et al.2011] Wayne Xin Zhao, Jing Jiang, Jiashu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. 2011. Comparing twitter and traditional media using topic models. In Advances in Information Retrieval, pages 338–349. Springer.