"Shall I Be Your Chat Companion?" Towards an Online Human-Computer Conversation System

Rui Yan[1,3]
[1]Institute of Computer Science and Technology (ICST)
Peking University
Beijing 100871, China
yanrui02@baidu.com

Yiping Song[2]
[2]Department of Computer Science and Technology
Peking University
Beijing 100871, China
songyiping@pku.edu.cn

Xiangyang Zhou[3]
[3]Baidu Inc.
No. 10 Xibeiwang East Road,
Beijing 100193, China
zhouxiangyang@baidu.com

Hua Wu[3]
[3]Baidu Inc.
No. 10 Xibeiwang East Road,
Beijing 100193, China
wu_hua@baidu.com

ABSTRACT

To establish an automatic conversation system between human and computer is regarded as one of the most hardcore problems in computer science. It requires interdisciplinary techniques in information retrieval, natural language processing, and data management, etc. The challenges lie in how to respond like a human, and to maintain a relevant, meaningful, and continuous conversation. The arrival of big data era reveals the feasibility to create such a system empowered by data-driven approaches. We can now organize the conversational data as a chat companion. In this paper, we introduce a chat companion system, which is a practical conversation system between human and computer as a real application. Given the human utterances as queries, our proposed system will respond with corresponding replies retrieved and highly ranked from a massive conversational data repository. Note that "practical" here indicates *effectiveness* and *efficiency*: both issues are important for a real-time system based on a massive data repository. We have two scenarios of single-turn and multi-turn conversations. In our system, we have a base ranking without conversational context information (for single-turn) and a context-aware ranking (for multi-turn). Both rankings can be conducted either by a shallow learning or deep learning paradigm. We combine these two rankings together in optimization. In the experimental setups, we investigate the performance between effectiveness and efficiency for the proposed methods, and we also compare against a series of baselines to demonstrate the advantage of the proposed framework in terms of p@1, MAP, and nDCG. We present a new angle to launch a practical online conversation system between human and computer.

Keywords

Human-computer conversation; big data; rank optimization

CIKM'16 , October 24-28, 2016, Indianapolis, IN, USA
© 2016 ACM. ISBN 978-1-4503-4073-1/16/10...$15.00
DOI: http://dx.doi.org/10.1145/2983323.2983360

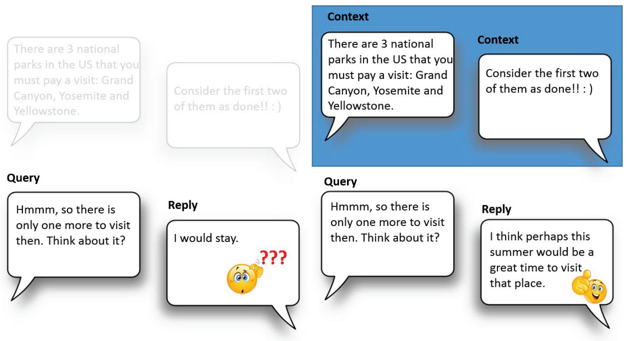
1. INTRODUCTION

To create a virtual assistant and/or chat companion system with adequate artificial intelligence has always been a long cherished goal for researchers and practitioners. It is believed to be challenging for computers to maintain a relevant, meaningful and continuous conversation with humans. How to respond like a human generally involves interdisciplinary techniques such as information retrieval, natural language processing, as well as big data management. A significant amount of efforts have been devoted to the research for decades, and promising achievements have been gradually achieved so that we can expect real applications in real life, rather than in Sci-Fi movies or research labs only.

The demand for virtual assistant/chat companion system leads to cutting-edge technology in the spotlight from both academia and industry. The arrival of big data era also accelerates the development of human-computer conversation studies. Owing to the public resources for conversations on the web, we are likely to learn what to reply given (almost) any inputs by retrieving from the conversational repository. It is probably a great timing to build such data-driven conversation systems between humans and computers. We are motivated to establish an online chat companion system for real-time services. Since there is a clear industry-driven background for our study, we ought to make the system practical, where "practical" means both *effectiveness* and *efficiency*. These two issues are fundamental for an online system.

Human-computer conversation systems have been evolving for years. Researchers have firstly investigated into task-oriented conversation systems [34, 40, 46]. To maintain conversations within a specified domain, it would be more feasible to create prior knowledge to handle specific tasks such as flight booking or bus route enquiry [23, 46]. One of the most obvious drawbacks for the domain-specific service is that the conversation cannot go beyond the domain of the system, and that the way to function is nearly impossible to be generalized to open domain. It is only recently that non-task-oriented dialogue, a.k.a. open-domain conversation, has been attracting the attention for its functional, social, and entertainment roles [3, 2, 26, 14, 39, 20, 32, 9, 19, 48]. In this paper, we set our target at creating a human-computer chat companion system (i.e., a ChatBot) in the open domain.

Building an open-domain ChatBot system to interact with humans is interesting but extremely challenging. Firstly, since people


There are 3 national parks in the US that you must pay a visit: Grand Canyon, Yosemite and Yellowstone.
Consider the first two of them as done!! :)
Query
Hmmm, so there is only one more to visit then. Think about it?
Reply
I would stay.
???
Context
There are 3 national parks in the US that you must pay a visit: Grand Canyon, Yosemite and Yellowstone.
Context
Consider the first two of them as done!! :)
Query
Hmmm, so there is only one more to visit then. Think about it?
Reply
I think perhaps this summer would be a great time to visit that place.

(a). Query-reply w/o contexts. (b). Query-reply w/ contexts.

Figure 1: Take a short multi-turn (2 turns) conversation for example. It would be quite confusing and less meaningful to select a reply to the query alone without using the conversation context. It is natural to select a reply about travel to the query when considering the context about travels in National Parks.

are free to say anything to the system, it is infeasible to prepare the knowledge for interaction before hand. Secondly, the possible combinations of conversation status is virtually infinity so that conventional hand-crafted rules would fail to work for unexpected human utterances [35]. As mentioned, the emerging increase of big web data can greatly advance the development of conversation systems in open-domain conversations. Owing to the diversity of the web, a retrieval-based system can retrieve at least some replies for almost any of the user input, and then returns a reply, which is a great advantage. To this end, we establish the conversation system based on retrieval-and-ranking based method.

In general, there are two typical scenarios for computers to understand conversations: 1) single-turn and 2) multi-turn conversation. A single-turn conversation may refer to the beginning of a (multi-turn) conversation, or a dialogue shift from the current conversation. A multi-turn conversation usually denote an on-going conversation which lasts for several turns. For multi-turn conversation, the series of conversation history can be utilized as additional information, namely "contexts". Single-turn conversations are easy to understand. No context information is available, nor necessary. We illustrate a simple multi-turn scenario in Figure 1, where context information is needed for the conversation. Based on the observations, we design two ranking mechanisms in particular to handle single-turn and multi-turn conversation in this paper, namely *base ranking* and *context-aware ranking*. We combine both rankings in optimization in order to suit both scenarios.

In this paper, we introduce a data-driven, real-time conversation system which provides chat service based on the tremendously large corpus from the Web. During each conversation session, users input a sequence of messages to "chat" with the computer, and the ChatBot will be able to return a corresponding series of replies. We organize functional components together into pipelines and apply timely efficient optimization to make the system framework practical.

To summarize, our contributions are as follows:

• The 1st contribution is that we introduce a straightforward yet **effective** rank optimization framework for the human-computer conversation system based on retrieval. The framework is general, and it adapts well for both single-turn and multi-turn conversation scenarios in open domain.

• The 2nd contribution is that we examine the **efficiency** issue for the proposed system to be practical. We examine the difference of *shallow learning* and *deep learning* based candidate rankings for both base ranking and context-aware ranking. We investigate the trade-off between effectiveness and efficiency.

We also run experiments to examine the *effectiveness* and *efficiency* of our proposed framework in comparison with several alternative methods. We are able to achieve the first-tier level for effectiveness using the deep learning based ranking, compared with state-of-the-art approaches. Using the shallow learning based ranking, we can choose to outperform almost all rivals in terms of running time, while remain a relatively high effectiveness performance. We provide a useful perspective of view to launch a practical system for online chat service.

In Section 2 we start by reviewing previous work. Then we introduce the conversation system framework, including pipeline components, ranking paradigm with optimized combination designs. We describe experiments and evaluations in Section 4, including experimental setups, performance comparisons and result discussions. Finally we draw conclusions in Section 5.

2. RELATED WORK

Early work on conversation systems is in general based on rules or templates and is designed for specific domains [34, 40]. These rule-based approaches require no data or little data for training, while instead require much manual effort to build the model, or to handcraft the rules, which is usually very costly. The conversation structure and status tracking in vertical domains are more feasible to learn and infer [46]. However, the coverage of the systems is also far from satisfaction. Later, people begin to pay more attention to automatic conversation systems in open domains [32, 9].

From specific domains to open domains, the need for a big amount of data is increasing substantially to build a conversation system. As information retrieval techniques are developing fast, researchers obtain promising achievements in (deep) question and answering system. In this way, an alternative approach is to build a conversation system with a knowledge base consisting of a number of question-answer pairs. Leuski *et al.* build systems to select the most suitable response to the current message from the question-answer pairs using a statistical language model in cross-lingual information retrieval [16], but have a major bottleneck of the creation of the knowledge base (i.e., question-answer pairs) [17]. Researchers propose augmenting the knowledge base with question-answer pairs derived from plain texts [24, 5]. The number of resource pairs in this way can be to some extent expanded but still relatively small while the performance is not quite stable either. Knowledge bases are important sources for better human language/text understanding [36, 37, 38]. It will be interesting to incorporate such knowledge into human-computer conversation.

Nowadays, with the prosperity of social media and other Web 2.0 resources, such as community question and answering (cQA) or microblogging services, a very large amount of conversation data becomes available [39]. A series of information retrieval based methods are applied on single-round short text conversation based on microblog data [20]. Higashinaka *et al.* also combine template generation with the search based methods [9]. Ritter *et al.* have investigated the feasibility of conducting short text conversation by using statistical machine translation (SMT) techniques, as well as millions of naturally occurring conversation data in Twitter [26]. In the approach, a response is generated from a model, not retrieved from a repository, and thus it cannot be guaranteed to be a legitimate natural language text.

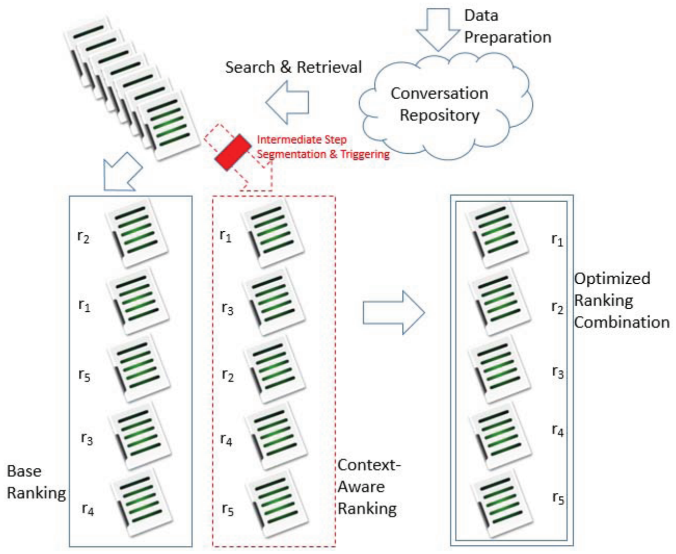


Figure 2: The illustration of the ChatBot system framework, including both off-line and online process. The candidates are ranked with and without contexts (if any) and combined both rankings in optimization.

Recently, with the fast development of deep learning techniques, efforts are devoted in the neural network-based conversation systems. A neural conversation model is proposed using a sequence-to-sequence manner [33]. Neural Responding Machine (NRM) is developed as a generative way to respond to human messages, using recurrent neural networks (RNN) [29]. The single-turn conversation generation is then extended to multi-turn conversation with conversational contexts are encoded for RNN [31]. Attention schema is also incorporated into the conversation model [45]. A hierarchical neural network model is proposed to model human conversations [28]. These generation-based neural networks cannot guarantee natural generations as well. What is more, neural conversation models tend to generate trivial or safe, commonplace responses (e.g., I don't know) regardless of the input [18]. Retrieval-based methods are more likely to find interesting candidate replies which are originally written by humans. Ji *et al.* introduce an information retrieval approach for short-text conversation using deep match metrics [14]. The architecture is further developed as a learning-to-match schema for short texts [20]. These methods are for single-turn conversations only, with no context information incorporated. In our previous work, we have proposed a deep learning-to-respond schema which incorporates contextual information for multi-turn conversations [43].

We can see these methods characterize new insights and formulate good models. Yet no previous studies examine the conversation system from a practical enterprise perspective, investigating the issues of *effectiveness* and *efficiency*. We hereby launch the ChatBot platform, and share the hands-on experience from an industry angel. We provide the real-time service, accessible for everyone.

3. SYSTEM FRAMEWORK

To create the conversation system, we need to establish a pipeline of components including data preparation, search and retrieval, and rankings with optimization. Roughly, we have an off-line process and an online process. We will go through these procedures in this section.

Table 1: An example of the original microlog post and its replies. We anonymize user information for privacy.

POSTING:
一把年纪的人居然近视了...求个眼镜做礼物! (It is unbelievable to have myopia at an "old" age... Wish a pair of glasses as my gift!)
REPLY 1: 我送给你! (I will offer one for you!)
REPLY 2: 可以恢复的，别紧张 (It can be recovered. Relax.)

Table 2: To create the conversation resource, we separate the post and replies as $\langle$*posting-reply*$\rangle$ pairs. In this way, we provide different ways to respond to a given query and make diversified conversations.

POSTING:	**POSTING:**
一把年纪的人居然近视了...求个眼镜做礼物! (It is unbelievable to have myopia at an "old" age... Wish a pair of glasses as my gift!)	一把年纪的人居然近视了...求个眼镜做礼物! (It is unbelievable to have myopia at an "old" age... Wish a pair of glasses as my gift!)
REPLY:	**REPLY:**
我送给你! (I will offer one for you!)	可以恢复的，别紧张 (It can be recovered. Relax.)

3.1 Off-line Process

Data Preparation. With the prosperity of Web 2.0, people interactively have conversations with each other on the Web. In Baidu Inc., there are trillions of webpages indexed in PB level of storage for Baidu search engine[1], which provides a huge thesaurus for conversation data. Therefore we are able to collect sufficient human conversation data taken from social media such as microblogging websites or forums, where users can publish a *posting* message visible to the public, and then receive a bunch of *replies* in response to their posting. Such conversations do not occur in strict real-time style but they are literally real human conversations. We collect the conversational data stored as $\langle$*posting-reply*$\rangle$ pairs. We show some of the examples in Table 1-2.

In the sample shown above, the first message of a conversation is typically unique, not directed at any particular user but instead broadcast to all audience. There are many flexible forms to reply to a given message, which is exactly the nature of real conversations: various response but all possibly appropriate, with different aspects of information to fulfill a conversation. We then separate the posting-replies into a group of *posting-reply* pairs, each with two utterances from the conversation. The dataset is demonstrated to be a perfectly rich resource to explore candidates to help completing a human-computer conversation in natural languages.

When the data is collected, we then pre-process the obtained resources for data cleaning. We filter the data by removing extremely short replies and those of low linguistic quality such as pointless babblings according to the evaluation framework put forward in [42, 44, 41], so as to maintain meaningful, high-quality conversation records. We also remove inappropriate conversations or incomplete sessions. We do not intend to include conversational resources with impolite, rude, or even dirty words. Besides, we also remove the conversation sessions with meaningless symbols, out-of-vocabulary words or pure numbers, which we believe they

[1]http://www.baidu.com

are less likely to constitute a good conversation. Besides, we also conduct stop word removal.

After data cleaning, we store the resource data on Baidu search platform and deploy a standard inverted indexing using terms using Baidu file systems and retrieval infrastructure, which supports regular updates (weekly or monthly) and incorporates new conversation resources. The data is incrementally bigger and bigger as time goes by. The procedures might take time but it is acceptable for an off-line process. Then the indexed documents in the ⟨*posting-reply*⟩ format will now be ready for online retrieval.

3.2 Online Process

Search and Retrieval. After taking in the user issued query, which could be one or more terms, we apply a standard retrieval process via keyword search on the conversation data resource using the light-weight search platform in Baidu Inc. Unlike the traditional search task deployed on general Baidu Search Engine which performs very heavy computing with thousands of features, the conversation based search is much lighter. We treat each pair of posting and reply as a short "virtual" document. Hence we incorporate minor adaption that now each *virtual document* to retrieve actually consists of a pair of *posting* and *reply*, even though we only need to return the reply as the output. The whole corpus is formatted in an inverted index prepared off-line. This retrieval process returns a list of all potentially relevant replies to the query from the corpus. With these candidate replies, we proceed to the next step.

Rankings with Optimization. We next conduct the relevance ranking procedure, to distinguish the highly relevant candidate replies from less relevant replies retrieved from the conversation archive. This step aims at matching postings and/or replies and then filtering down the candidate pool. To optimize the ranking, the ranking paradigm will need to be general enough so as to tackle every scenario of human-computer conversation no matter when 1) the user initiates a new conversation, 2) the user continues an on-going conversation and 3) the user continues the conversation with new content shifts in the meanwhile. These situations can be mapped into two scenarios, either single-turn conversation, or multi-turn conversation with contexts. Correspondingly, we design two rankings namely *base ranking* for single-turns and *context-aware ranking* for multi-turns. We also introduces an intermediate component to segment the conversation sessions, so as to decide what to use as the contexts and how to use contexts: we ought to use a succession dialogue as the contexts. With all these metrics, we are able to optimize the candidate reply selection with both ranking schemes. More importantly, since we provide real-time conversation service, we need to be practical about context utilization. It is natural that as context grows, the cost to use contexts will grow as well. We will manage to make the cost remain (almost) the same, rather than let the cost grow linearly or exponentially as context length grows.

Since the rank optimization is a core component in the retrieval based conversation system and mixed with strategies, we next proceed to elaborate the ranking with optimization.

4. RANKING WITH OPTIMIZATION

4.1 Problem Formulation

The research problem of automatic human-computer conversation is defined as one or more turns of chattings between man and computer. Within each turn of the conversation, given the message issued from human, the computer would provide a reply in response to the coming message. Given the user messages as queries, our system retrieves related replies from a vast repository of conversation data and returns the most appropriate response. We propose to suit both scenarios of single-turn and multi-turn conversation using a base ranking as well as a context-aware approach, and expect the conversation system can respond like a human. We have the following definitions in Table 3.

Table 3: Different formulations for single-turn conversation and multi-turn conversation with context information.

	Single-Turn	Multi-Turn
Inputs.	q_n	$q_n, \mathbb{C}$
Outputs.	r_n	r_n
Obj. Func.	$r_n^\star = \underset{r_n}{\text{argmax}}\, f(r_n \vert q_n)$	$r_n^\star = \underset{r_n}{\text{argmax}}\, f(r_n \vert q_n, \mathbb{C})$

We have a conversation resource in ⟨*posting-reply*⟩ pairs. For a single-turn conversation, the situation is simple and clear. Given a particular query q, we ought to return the selected reply r. In a multi-turn conversation scenario, for the reply r_n in the n-th turn, a single-turn system utilizes only the n-th query q_n while a multi-turn system also includes a successive chatting history as contexts $\mathbb{C}$=$\{q_1, r_1, \ldots, q_{n-1}, r_{n-1}\}$. $f(.)$ is a certain scoring metric to indicate the tendency to choose r as output for the given inputs. The metric will be discussed in details in the following sections. We optimize the best reply $r_n^\star$ given the context $\mathbb{C}$.

4.2 Rankings

We conduct a ranking to distinguish highly relevant candidate replies from less relevant replies retrieved from the conversation archive. This step aims at matching postings and/or replies and then filtering down the candidate pool. We have both base ranking (i.e., single-turn without contexts) and context-aware ranking during this step. We are able to obtain two ranking scores for each of the candidate replies. Since we propose to utilize the context information, we ought to at first select contexts since we only use a succession conversation associated with q as the contexts $\mathbb{C}$.

• **Context Selection.** Firstly, we will address the context selection issue. The intuition is that given the whole conversation session, not all of the texts are successive with the query, neither should the additional information be incorporated. We hereby use an intermediate step to decide when and what session texts to be utilized as "contexts". We deploy a TextTiling-like [30, 7, 8] segmentation method with dialogue act to segment dialogue "sessions" in the conversation, and we use a succession dialogue associated with q as the contexts. The session segmentation ensures that we use appropriate contexts as additional information.

• **Base Ranking.** We first offer a base ranking for the retrieved replies, which is to distinguish the highly relevant candidate replies from the irrelevant responses retrieved from the conversation archive. This step aims at matching postings and/or replies and then filtering down the candidate pool through semantic relevance measurement and a series of other evaluation metrics. Given a query q, we obtain a ranking list τ of the top-k ranked replies $\mathcal{R}_q = \{r_1, r_2, \ldots, r_k\}$ according to the relevance score given by the ranker. A permutation (or ranking) τ is generated in the descending order. Each r_i is represented as a vector x_i where each $\mathrm{x}_i \in \mathbb{R}^{|\mathrm{x}|}$ has $|\mathrm{x}|$ dimensions. A ranking function $\mathbb{R}^{|\mathrm{x}|} \to \mathbb{R}$ is trained and applied to rank the replies, and $\tau(r_i) < \tau(r_j)$ means r_i is more preferred over r_j in the ranking list (a.k.a., $f(r_i|q) > f(r_j|q)$ since it is based on single turns). We have two instinctively different ranking paradigms based on shallow learning and deep learning. The shallow/deep rankers will be described in Section 4.3. Through this process, we are able to obtain a ranking score for each of the candidate replies and then we rank the replies accordingly. This ranking list is regarded as a base ranking using q only. We omit the subscript of q and use $\mathcal{R}$ to represent $\mathcal{R}_q$ when there is no ambiguity.

Till now, we only consider the information without contextual information to match the candidate replies to the query, and hence create the base ranking τ_b. However, the context-insensitive information might not be sufficient enough, especially under the scenario of multi-turn conversation when we have a conversation history to evaluate which reply would be better given a particular context. We next incorporate the context-aware ranking.

• **Context-Aware Ranking.** Now we aim at examining which candidate reply is more appropriate given the context under a multi-turn conversation scenario. We also have a vector-based context representations to match with the top-ranked candidate replies, and hence refine the rankings. Likewise, we use a shallow ranker or a deep ranker.

After the base ranking and the context-aware ranking, we opt to combine both dimensions of rankings together in order to optimize the final ranking orders for all candidate replies. Then, the conversation system selects the reply with the highest rank as the output to respond. Next, we will further go through the shallow and deep rankers, and then the details of rank combination and optimization.

4.3 Shallow Ranker v.s. Deep Ranker

In this section, we introduce two ranking methods for candidate replies, using shallow learning and deep learning correspondingly.

4.3.1 Shallow Learning

For the shallow ranker, we deploy a Maximum Entropy classifier [13] to offer the base ranking and the context-aware ranking for the retrieved replies. The classifier utilizes a series of features. At first we introduce some shallow representations of the documents, i.e., queries, replies, and postings.

Term-based Representation. Most intuitively, we represent the texts by the vector space model [21]. In this way, the content of the context can be represented as $c(d) = \langle\pi(w_1, d), \pi(w_2, d), \ldots, \pi(w_{|d|}, d)\rangle$ where $\pi(w_i, d)$ denotes the weight for term w_i which occurs in the text d. For the conversation data, we evaluate the term weighting using the standard *tf-idf* measurement, which reflects term importance [21]. We calculate the cosine similarity between two term vectors.

Topic-based Representation. "Topics" have long been investigated as the abstractive semantic representation [10]. We apply the unsupervised Latent Dirichlet Allocation [4] to discover topics. We obtain the probability distribution over topics for the text pieces. The inferred topic representation is the probabilities for the piece of text belonging to a certain topic. We empirically train a 1000-topic model and represent texts as topic vectors. We also calculate the cosine similarity between two vectors.

Entity-based Representation. Named entities are a special form of terms. In this study, we distinguish *persons*, *locations* and *organizations* from plain texts with the help of named entity recognition techniques and maintain a vector of recognized entities for the conversation. Based an established knowledge graph mined from Baidu search logs, we can calculate the similarity (measured by entity distance in the knowledge graph) between two entity-based vector representations.

We formulate the following pairwise features. For pointwise features, we basically calculate a matching score, either by similarity using the standard cosine metric, or by dependency measurement using the mutual information [21] metric.

Query-Reply Matching Score. We calculate the matching score between the query and the candidate reply by 1) cosine similarity and 2) mutual information based on the mentioned representations.

Context-Reply Matching Score. Similarly, we calculate the matching score between the context and the candidate reply, calculated by the matching metrics based on the three shallow representations.

Query-Posting Matching Score. We also consider the matching score between query and the original posting associated with the candidate reply in the $\langle$*posting-reply*$\rangle$ pairs, calculated by the matching metrics.

Context-Posting Matching Score. Again, we calculate the matching score between the context and the original posting associated with the candidate reply, calculated by the mentioned matching metrics.

Statistical Machine Translation. Statistical machine translation (SMT) is a machine translation paradigm which translates one sentence in a language to a sentence in another language. If we treat queries/contexts and replies as different languages, we can train a translation model to "translate" queries/contexts into replies based on the training corpora. Note that for the SMT score, we only calculate based on term-based representation.

We also include pointwise features as follows:

Language Model. Generative score from high quality language model for the candidate reply. The language model is trained based on large news data. We train the language model in unigram based on millions of news articles archived from various mainstream news websites.

Average Term Weighting. We calculate the average *tf-idf* score for the reply as the weight. A candidate reply with higher weights is more likely to be importance.

Length. This feature denotes the length of replies. Too short replies are not preferred. We conduct a normalization to map the value to [0,1].

Fluency. Fluency is to examine whether two neighboring terms have a large co-occurrence likelihood. We calculate the co-occurrence probability for the bi-grams of the candidate reply and then take the average value as the fluency.

We feed all the calculated features of a candidate reply into the scoring function and rank the reply accordingly. The features are empirically hand-crafted and the learning is shallow. In contrast to the shallow learning ranker, we next propose a deep learning ranker.

4.3.2 Deep Learning

The deep ranker based on deep learning techniques do not rely on hand-crafted features from empirical expertise. In recent years, deep neural networks (DNNs, also known as *deep learning*) have made significant improvement. With big data available, DNNs are highly automated learning machines; they can extract underlying abstract features of data automatically by exploring multiple layers of non-linear transformation [1].

As mentioned, we have two parts, i.e., $\langle$*posting-reply*$\rangle$, to compare with the query and/or the context. The scoring function outputs a scalar in $\mathbb{R}$ (appropriateness or inappropriateness) in for a particular candidate reply, given either the query itself (i.e., the base ranking part) or the context (i.e., the context-aware ranking part). Both the base ranking score and the context-aware ranking score are computed by the same deep neural network architecture, but their parameters are different so that the scoring functions can depict different meanings. In particular, the deep structure for sentence pair modeling includes the following components.

Word Embeddings. Traditional models usually treat a word as a discrete token; thus, the internal relation between similar words would be lost. Word embeddings [22] are a standard apparatus in neural network-based text processing. A word is mapped to a low dimensional, real-valued vector. This process, known as vectorization, captures some underlying meanings. Given enough data, usage, and context, word embeddings can make highly accurate guesses about the meaning of a particular word. Embeddings can equivalently be viewed that a word is first represented as a one-hot vector and multiplied by a look-up table [22].

In our model, we first vectorize all words using their embeddings, which serve as the foundation of our deep neural networks. Word embeddings are initialized randomly, and then tuned during training as part of model parameters.

Bi-Directional LSTM. We use a bi-directional long short term memory (Bi-LSTM) recurrent network to propagate information along the word sequence. A recurrent neural network (RNN) keeps a hidden state vector, which changes according to the input in each time step. As RNNs can iteratively aggregate information along a sequence, they are naturally suitable for sentence modeling.

LSTM is an advanced type of RNN by further using memory cells and gates to learn long term dependencies within a sequence [33, 25]. LSTM models are defined as follows: given a sequence of inputs, an LSTM associates each position with *input*, *forget*, and *output gates*, denoted as i_t, f_t, and o_t respectively. The vector l_t is used to additively modify the memory contents. Given an input sentence $S = \{x_0, x_1, \ldots, x_T\}$, where x_t is the word embedding at position t in the sentence. LSTM outputs a representation h_t for position t, given by

$$\begin{aligned}
\begin{bmatrix} i_t \\ f_t \\ o_t \\ l_t \end{bmatrix} &= \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot \begin{bmatrix} h_{t-1} \\ e_t \end{bmatrix} \\
\tilde{h}_t &= f_t \cdot \tilde{h}_{t-1} + i_t \cdot l_t \\
h_t^s &= o_t \cdot \tilde{h}_t
\end{aligned} \tag{1}$$

where $\tilde{h}$ is an auxiliary variable and can be viewed as the information stored in memory cell. $\sigma(\cdot) = \frac{1}{1+e^{-\cdot}}$ is a known as a sigmoid/logistic function.

A single directional LSTM typically propagates information from the first word to the last; hence the hidden state at a certain step is dependent on its previous words only and blind of future words. The variant Bi-LSTM [6] is proposed to utilize both previous and future words by two separate RNNs, propagating forward and backward, and generating two independent hidden state vectors $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$, respectively. The two state vectors are concatenated to represent the meaning of the t-th word in the sentence, i.e., $h_t = \left[\overrightarrow{h_t}; \overleftarrow{h_t}\right]$.

Convolution. We further apply a convolutional neural network (CNN) to extract local neighboring features of successive words—i.e., discriminative word sequences can be detected—yielding a more composite representation of the sentences. The structure of CNN in this work is similar to [15]. Unlike RNNs, CNNs only impose local interactions between successive words within a filter (size m).

Concretely, we build a CNN upon the output of Bi-LSTM. For every window with the size of m in Bi-LSTM output vectors, i.e., $(H_t)_m = [h_t, h_{t+1}, \cdots, h_{t+m-1}]$, where t is a certain position, the convolutional filter $F = [F(0), \ldots, F(m-1)]$ will generate a vector sequence using the convolution operation "$*$" between the two vectors. More formally, the convolution results in a vector where each component is as follows:

$$o_F = \tanh\left[\sum_{i=0}^{m-1} h(t+i) * F(i)\right] \tag{2}$$

In practice, we also add a scalar bias b to the result of convolution. In this way, we obtain the vector o_F is a vector, each dimension corresponding to each word in the sentence.

Notice that the above equation describes a single "slice" of convolution. In fact, we may have multiple feature filters and thus multiple feature maps. Different filters do not share parameters (F and b), so that they can capture different meanings.

Pooling, Concatenation, and Matching. On the basis of sentence representations using Bi-LSTM with CNN, we can model the interactions between two sentences. We apply pooling to aggregate information along the word sequence. In particular, a max pooling layer chooses the maximum value in each dimension in the feature maps after the convolution, indicating how much the feature is most satisfied along the sequence.

We have two matching matrice between posting-query (or context) and reply-query (or context) using standard cosine similarity. Hence we have two scalar scores for the matching matrice. We concatenate these individual sentences' vector representations (i.e., query/context, posting, and reply) with the scores from the matching matrice. Then the concatenated vectors are fed to an ensuing network for further information mixing. Vector concatenation for sentence matching is also applied in other studies like [47], which is effective yet of low complexity order, compared with other word-by-word matching [11], or attention methods [27].

The joint vector is then passed through a 3-layer, fully-connected, feed-forward neural network, also known as *multi-layer perception* (MLP) [1], which allows rich interactions between a sentence pair from one of the three components. The network enables to extract features automatically, starting from lower-level representations to higher-level ones. Finally, a single neuron outputs the score between a query (or the context) and a reply. The final scoring neuron is essentially a linear regression.

4.4 Rank with Optimization

Given the two ranking lists (i.e., base ranking and context-aware ranking), our proposed system applies an optimized combination of the ranked lists. We optimize the combination so that the refined rank should not deviate too much from either of the two rankings. Then, the system selects the reply with the highest ranking as the output to respond.

We do not directly add the base ranking scores and context-aware ranking scores together as the measurement to calculate the final rankings. Let τ_b be the base ranking and τ_c be the context-aware ranking, we need to estimate the final ranking as τ, and also estimate the rank $\tau(r_i)$ for each candidate reply r_i. We aim at optimizing the following objective cost function O(τ),

$$\begin{aligned}
O(\tau) =& \alpha \sum_{i=1}^{|\tau|} \mathcal{B}_i \big(||\frac{\tau(r_i)}{\Psi_i}|| - ||\frac{\tau_b(r_i)}{\mathcal{B}_i}||\big)^2 \\
&+ \beta \sum_{i=1}^{|\tau|} \mathcal{C}_i \big(||\frac{\tau(r_i)}{\Psi_i}|| - ||\frac{\tau_c(r_i)}{\mathcal{C}_i}||\big)^2
\end{aligned} \tag{3}$$

where $\mathcal{B}_i$ is the base ranking score while $\mathcal{C}_i$ is the context-aware ranking score. Ψ_i is expected to be the merged ranking score, namely the "appropriateness", which will be defined later. Among the two components in the objective function, the first component means that the refined rank should be close to the base rank. We use $(||\frac{\tau(r_i)}{\Psi_i}|| - ||\frac{\tau_b(r_i)}{\mathcal{B}_i}||)^2$ instead of $(\tau(r_i) - \tau_b(r_i))^2$ in order to distinguish the detailed differences associated with scores and ranks. The second component is similar by making rank close to context-aware rank.

Our goal is to find $\tau(r_i) = \tau(r_i^\star)$ to minimize the cost function, i.e., τ = argmin $O(\tau)$. τ is the final rank merged by our algorithm. To minimize $O(\tau)$, we compute its first-order partial derivatives.

$$\frac{\partial O(\tau)}{\partial \tau(r_i)} = \frac{2\alpha}{\Psi_i}(\frac{\mathcal{B}_i}{\Psi_i}\tau(r_i) - \tau_b(r_i)) + \frac{2\beta}{\Psi_i}(\frac{\mathcal{C}_i}{\Psi_i}\tau(r_i) - \tau_c(r_i)) \quad (4)$$

Let $\frac{\partial O(\tau)}{\partial \tau(r_i)} = 0$, we get

$$\tau(r_i^\star) = \frac{\alpha\Psi_i\tau_b(r_i) + \beta\Psi_i\tau_c(r_i)}{\alpha\mathcal{B}_i + \beta\mathcal{C}_i} \quad (5)$$

Two special cases are that if (1) $\alpha = 0, \beta \neq 0$: indicating we only use the base ranking, i.e., single-turn without contexts. (2) $\alpha \neq 0$, $\beta = 0$, indicating we only use context-aware ranking without base ranking.

We define Ψ_i as the weighted combination of merged ranking scores from both rankings:

$$\Psi_i = \frac{\alpha\mathcal{B}_i + \beta\mathcal{C}_i}{\alpha + \beta} \quad (6)$$

In this way, final Ψ_i is dependent on base ranking score and the additional context-aware ranking score. We plug Equation (6) into Equation (5), and obtain a more concise format with no ranking score of Ψ: $\tau(r^\star)$ is a weighted combination of base ranking and context-aware ranking by:

$$\tau(r_i^\star) = \frac{\alpha}{\alpha + \beta}\tau_b(r_i) + \frac{\beta}{\alpha + \beta}\tau_c(r_i) \quad (7)$$

The final ranking equation indicates an efficient calculation approach with an analytical solution. Even if the contexts are consisted by several utterances, we just need to process all the contexts together as a batch, rather than a sentence-by-sentence style, which makes the service efficiently practical.

5. EXPERIMENTS AND EVALUATION

In this section, we evaluate the proposed system for conversation task against a series of baselines based on the huge conversation resource. The objectives of our experiments are to 1) evaluate the effectiveness of our proposed framework, 2) evaluate the efficiency issue of different ranking approaches, and 3) investigate the trade-off between effectiveness and efficiency.

5.1 Experimental Setups

5.1.1 Dataset

As mentioned, we collected massive conversation resources from various forums, microblog websites, and cQA platforms including Baidu Zhidao[2], Douban forum[3], Baidu Tieba[4], Sina Weibo[5], etc. The dataset is the same dataset that we used in our previous work [43]. In total, the database contains ~10 million $\langle$*posting*, *reply*$\rangle$ pairs. Some statistics are summarized in Table 4.

We constructed the dataset of 1,606,583 samples to train the deep neural networks, 357,018 for validation, and 11,097 for testing. It is important that the dataset for learning does not overlap with the database for retrieval, so that we strictly comply with the machine learning regime. For each training and validation sample, we randomly chose a reply as a negative sample. Validation was based on the accuracy of positive/negative classification. For the test set, we hired workers on a crowdsourcing platform to judge the appropriateness of 30 candidate replies retrieved for each query. Each sample was judged by 7 annotators via majority voting based on the *appropriateness* for the response given the query and contexts (if any): "1" denotes an appropriate response and "0" indicates an inappropriate one.

Table 4: Data statistics. Postings and replies are all unique.

Source	#Posting	#Reply	#Vocabulary
Zhidao	8,915,694	3,705,302	1,499,691
Douban	10,618,981	2,963,226	483,846
Tieba	4,189,160	3,730,248	1,046,130
Weibo	186,963	393,654	119,163
Misc.	3,056	1,548	4,297
Total	9,023,854	7,293,978	2,857,378

5.1.2 Hyperparameters

In our proposed model, we used 128-dimensional word embeddings, and they were initialized randomly and learned during training. As our dataset is in Chinese, we performed standard Chinese word segmentation. We maintained a vocabulary of 177,044 phrases by choosing those with more than 2 occurrences.

The bi-directional LSTM has 128 hidden units for each dimension; CNN is 256 dimensional with a window size of 3. We used stochastic gradient descent (with a mini-batch size of 100) for optimization, gradient computed by standard back-propagation. Initial learning rate was set to 0.8, and a multiplicative learning rate decay was applied. The above parameters were chosen empirically. We used the validation set for early stopping.

5.1.3 Evaluation Metrics

Given the ranking lists (annotated by crowdsourced workers) for test queries, we evaluated the performance in terms of the following metrics: precision@1 (p@1), mean average precision (MAP) [32], and normalized discounted cumulative gain (nDCG) [12]. Since the system outputs the best selected reply, p@1 is the precision at the 1st position, and should be the most natural way to indicate the fraction of suitable responses among the top-1 reply retrieved. Besides, we also provided the top-k ranking list for the test queries using nDCG and MAP, which test the potential for a system to provide more than one appropriate responses as candidates. We aimed at selecting as many appropriate responses as possible into the top-k list and rewarding methods that return suitable replies on the top.

Formally, the metrics are computed as follows.

$$\text{nDCG@i} = \frac{1}{|\mathcal{T}|}\sum_{\text{q}\in\mathcal{T}}\frac{1}{\text{Z}}\sum_{i=1}^{k}\frac{2^{r_i}-1}{\log(1+i)}$$

where $\mathcal{T}$ indicates the testing query set, k denotes the top-k position in the ranking list, and Z is a normalization factor obtained from a perfect ranking. r_i is the relevance score for the i-th candidate reply in the ranking list (i.e., 1: appropriate, 0: inappropriate).

MAP is computed by

$$\text{MAP} = \frac{1}{|\mathcal{T}|}\sum_{\text{q}\in\mathcal{T}}\frac{1}{N_q}\sum_{i=1}^{k}P_i \times r_i$$

Here N_q is the number of appropriate responses selected, and P_i is the precision at i-th position for the query.

[2]http://www.zhidao.baidu.com
[3]http://www.douban.com
[4]http://www.tieba.baidu.com
[5]http://www.weibo.com

Table 5: Overall performance comparison. For *Microsoft XiaoIce* and generative methods, we do not have MAP or nDCG scores since we generate one candidate reply for each query. Note that for the efficiency of training and testing queries, we report the time cost *per query*.

	Effectiveness Metrics				Efficiency (in milliseconds)	
System	p@1	MAP	nDCG@10	nDCG@20	Training	Testing
SMT (Ritter et al., [26])	0.363	—	—	—	37.8	0.4
NRM (Shang et al., [29])	0.465	—	—	—	185.3	4.2
LSTM-RNN (Sutskever et al., [33])	0.501	—	—	—	253.9	5.1
Microsoft XiaoIce	0.533	—	—	—	—	≈1.1
Random Match	0.266	0.246	0.289	0.353	—	≈0.0
Okapi BM25	0.272	0.253	0.302	0.368	—	≈0.0
DeepMatch (Lu and Li, [20])	0.457	0.317	0.454	0.508	193.8	3.9
LSTM-RNN (Palangi et al., [25])	0.338	0.283	0.371	0.431	221.6	4.8
ARC-CNN (Hu et al., [11])	0.394	0.294	0.421	0.477	165.3	3.5
DL2R (Yan et al., [43])	0.731	0.416	0.682	0.717	396.5	6.8
ROCF in DeepMatch	0.668	0.406	0.628	0.691	193.8	3.9
ROCF in LSTM-RNN	0.631	0.394	0.616	0.672	221.6	4.8
ROCF in ARC-CNN	0.647	0.401	0.608	0.661	165.3	3.5
ROCF-ShallowRanker	0.593	0.387	0.602	0.658	2.7	0.1
ROCF-DeepRanker	0.711	0.412	0.666	0.702	250.8	5.0

5.2 Algorithms for Comparison

To illustrate the performance of our approach, we include several alternative algorithms as baselines for comparison. The baselines can be divided into two categories, i.e., 1) generation-based methods and 2) retrieval-based methods for conversation systems from very recent studies. Since our proposed approach is technically a retrieval-based method, we mainly focus on the second category. For fairness we conducted the same pre-processing procedures and data cleaning for all algorithms.

Generation-based Conversation. For this group of algorithms, the conversation system will generate a response from a given input, i.e., a query from the user under the conversational scenario.

• *Statistical Machine Translation (SMT)*: SMT is a machine translation paradigm which translates one sentence in the source language to a sentence in the target language. If we treat queries and replies as separate languages, we can train a translation model to "translate" queries into replies. We implemented the phrase-based translation idea for conversation proposed in [26].

• *LSTM-RNN*: LSTM-RNN is basically a Recurrent Neural Network (RNN) using the Long Short Term Memory (LSTM) architecture. The RNN with LSTM units consists of memory cells in order to store information for extended periods of time. We use LSTM-RNN for both generation and retrieval baselines. For generation, we first use an LSTM-RNN to encode the input sequence (query) to a vector space, and then use another LSTM-RNN to decode the vector into the output sequence (reply) [33]; for retrievals, we adopt the LSTM-RNN to construct sentence representations and use cosine similarity to output the matching score [25].

• *Neural Responding Machine.* We implement the neural responding machine (NRM) proposed in [29], which is an RNN-based generation approach with a global-local attention schema.

Retrieval-based Conversation. The approaches within this group of baselines are based on retrieval systems, which return the best matched candidate reply out of the conversational repository given a particular query. Since our approach is retrieval-based, we select strong retrieval-based methods to make a thorough comparison.

• *Random Match.* The method randomly selects replies from the retrieved list for each query. Be aware it is not *true* random because it only randomizes the order of the retrieved results. The true random match is too weak to be included as a decent baseline.

• *Okapi BM25.* We include the standard retrieval technique to rank candidate replies. For each query, we retrieve the most relevant reply using BM25 model [21] from the corpus.

• *DeepMatch.* The DeepMatch method considers multiple granularity from the perspective of topics, obtained via LDA [20].

• *ARC-CNN.* The ARC approach is a CNN based method with convolutionary layers which construct sentence representations and produce the final matching scores via a MLP layer [11].

• *Deep Learning-to-Respond (DL2R).* We propose the DL2R system based on a query reformulation and result list merging framework in [43]. DL2R is also a general approach for both single-turn and multi-turn conversation scenarios.

• *Industry Application (Microsoft XiaoIce).* There are several industrial companies researching into automatic human-computer conversation as well, such as Microsoft Cortana, Google Now, Apple Siri and Huawei Noah. Since we develop the entire system based on Chinese corpus, it is intuitive that we compare with the Microsoft Chinese ChatBot named XiaoIce[6] (小冰), which could to some extent represent the state-of-the-practice conversation software from industry. Due to limited open API access, we can only obtain the top-1 reply from XiaoIce for each query.

• *Rank Optimized Conversation Framework (ROCF).* We propose ROCF using context information, which aims at retrieving more appropriate replies based on conversational contexts from previous turns, which is an optimized combination of base ranking and context-aware ranking. Since the proposed ROCF is an adaptive framework, we have two substitutive components of both shallow ranker, deep ranker and other rankers for the ranking part.

5.3 Results

Overall performance results are shown in Table 5. From the results, we have some interesting observations.

For the **effectiveness** concern, the performance of the generative methods is quite moderate, which concurs the judgment from [18, 29]. The automatic conversation generators tend to produce universal, trivial and ambiguous replies which are likely to answer a wide

[6]http://www.msxiaoice.com/

range of queries, but not specific enough to conduct a meaningful conversation. Besides, despite of the relatively high p@1 score for these methods, they do not naturally provide more than one candidate replies. In general, we observe that generative approaches using deep learning (i.e., NRM and LSTM-RNN) perform better than that without deep learning techniques.

For retrieval methods, it is easy to obtain a ranked list of candidate replies, which show great potential to conduct conversations with diversity. As to the retrieval-based methods, RandomMatch is a lower bound for all baselines. As we mentioned, it randomizes the order of the retrieved results. Hence the result is comparable to that of BM25, slightly worse. Okapi BM25 represents the standard (and simple) retrieval system. The performance for BM25 is not as good as the other deep learning-based retrieval systems, which is not surprising. Deep learning systems are proved to have strong capabilities to learn the abstractive representation [20, 29, 31]. The best performance is achieved by DL2R which is proposed in our previous work [43]. In the meanwhile, DL2R is the most time consuming method. The performance of deep learning based algorithms in general overwhelms that of shallow learning based algorithms. The DL2R method has the advantage of fine-grained contextual modeling.

Note that, our proposed ROCF-DeepRanker outperforms all other baselines, and achieves comparable (slightly weaker) result with DL2R. The ROCF-ShallowRanker outperforms the general matching algorithms with deep learning techniques, due to the conversational scenario formulation. Still, ROCF-ShallowRanker is not as effective as other deep learning metrics in ROCF framework. It is natural for the shallow representation of feature engineering. The observation is somehow meaningful: the proposed ROCF framework is extensive and compatible for different rankers.

Another minor issue is that we include the industrial application of Microsoft XiaoIce due to the industry background of our study. Since we cannot get access to the internal API of XiaoIce, there could be some network delay and hence the testing time is inaccurate. Still we outperforms XiaoIce in terms of p@1 metric, which could be regarded as one of the up-to-date industrial systems.

Since we target at launching a practical online system to provide real-time service, time **efficiency** is also a major concern. In general, deep learning methods require big data for model training, the advantage of our proposed approach becomes rather prominent. The time for training the deep learning models (either the generative method, or the retrieval method DeepMatch) was several magnitudes higher compared to the straightforward shallow learning models according to the training time and testing time. It is intuitive that measurements on terms and high-level features will be much faster than a series of heavily-computed calculations: embedding, convolution, and pooling, etc. Although the fastest method is the random match, the proposed ROCF-ShallowRanker seems to yield the best trade-off between effectiveness and efficiency for a practical system. Given enormous number of conversations happened every day, the time difference between shallowing learning and deep learning is greatly amplified. However, we still have great potential to increase the effectiveness performance when we switch to the deep learning way online. The proposed ROCF is rather adaptive and flexible.

6. CONCLUSION

In this paper, we propose a practical, which means *effective* and *efficient*, framework for human-computer conversation in open domain, which targets at providing real-time services. Given the user issued message as queries, our proposed system will be able to return a corresponding reply retrieved from a massive data repository ($\sim$10 million $\langle$*posting-reply*$\rangle$ pairs) to respond the human utterance, fundamentally based on an information retrieval framework given a vast conversation resource. The system suits both single-turn and multi-turn conversation. We start from a base ranking without context information, and then incorporate a context-aware ranking (if any) in an optimized combination. Performance comparisons between our proposed method against baselines demonstrate the advantage of our ROCF based on the rank optimization and combination approach. The deep learning methods are competitive with better effectiveness scores but have significant drawbacks in efficiency. The ShallowRanker achieves slightly weaker effectiveness but much faster in a timely manner. We provide another perspective of view to establish a practical conversation system. Yet, we still have plenty room to make improvement using ROCF-DeepRanker.

7. ACKNOWLEDGMENTS

This work is supported by the National High Technology R&D Program of China (Grant No. 2015AA015403) and National Basic Research Program of China (Grant No. 2014CB340505). We thank all the reviewers for their valuable comments, and thank the support from the Deep Learning Platform in Baidu Natural Language Processing Department.

8. REFERENCES

[1] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[2] F. Bessho, T. Harada, and Y. Kuniyoshi. Dialog system using real-time crowdsourcing and twitter large-scale corpus. In *SIGDIAL '12*, pages 227–231, 2012.

[3] T. W. Bickmore and R. W. Picard. Establishing and maintaining long-term human-computer relationships. *ACM Trans. Comput.-Hum. Interact.*, 12(2):293–327, June 2005.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.

[5] G. Cong, L. Wang, C.-Y. Lin, Y.-I. Song, and Y. Sun. Finding question-answer pairs from online forums. In *SIGIR '08*, pages 467–474, 2008.

[6] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Proc. Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.

[7] M. A. Hearst. Multi-paragraph segmentation of expository text. In *ACL '94*, pages 9–16, 1994.

[8] M. A. Hearst. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Comput. Linguist.*, 23(1):33–64, Mar. 1997.

[9] R. Higashinaka, K. Imamura, T. Meguro, C. Miyazaki, N. Kobayashi, H. Sugiyama, T. Hirano, T. Makino, and Y. Matsuo. Towards an open domain conversational system fully based on natural language processing. In *COLING'14*, 2014.

[10] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1-2):177–196, 2001.

[11] B. Hu, Z. Lu, H. Li, and Q. Chen. Convolutional neural network architectures for matching natural language sentences. In *NIPS*, pages 2042–2050, 2014.

[12] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, Oct. 2002.

[13] E. T. Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

[14] Z. Ji, Z. Lu, and H. Li. An information retrieval approach to short text conversation. *CoRR*, abs/1408.6988, 2014.

[15] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[16] A. Leuski, R. Patel, D. Traum, and B. Kennedy. Building effective question answering characters. In *SIGDIAL'09*, pages 18–27, 2009.

[17] A. Leuski and D. Traum. Npceditor: Creating virtual human dialogue using information retrieval techniques. *AI Magazine*, 32(2):42–56.

[18] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan. A diversity-promoting objective function for neural conversation models. 2016.

[19] X. Li, L. Mou, R. Yan, and M. Zhang. Stalematebreaker: A proactive content-introducing approach to automatic human-computer conversation. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2845–2851, 2016.

[20] Z. Lu and H. Li. A deep architecture for matching short texts. In *NIPS'13*, pages 1367–1375, 2013.

[21] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. 2008.

[22] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv:1301.3781*, 2013.

[23] M. Nakano, N. Miyazaki, N. Yasuda, A. Sugiyama, J.-i. Hirasawa, K. Dohsaka, and K. Aikawa. Wit: A toolkit for building robust and real-time spoken dialogue systems. In *SIGDIAL '00*, pages 150–159.

[24] E. Nouri, R. Artstein, A. Leuski, and D. R. Traum. Augmenting conversational characters with generated question-answer pairs. In *AAAI Fall Symposium: Question Generation*, 2011.

[25] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward. Deep sentence embedding using the long short term memory network: Analysis and application to information retrieval. *arXiv preprint arXiv:1502.06922*, 2015.

[26] A. Ritter, C. Cherry, and W. B. Dolan. Data-driven response generation in social media. In *EMNLP '11*, pages 583–593, 2011.

[27] T. Rocktäschel, E. Grefenstette, K. M. Hermann, T. Kočiskỳ, and P. Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[28] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. 2016.

[29] L. Shang, Z. Lu, and H. Li. Neural responding machine for short-text conversation. In *ACL-IJCNLP*, pages 1577–1586, 2015.

[30] Y. Song, L. Mou, R. Yan, L. Yi, Z. Zhu, X. Hu, and M. Zhang. Dialogue session segmentation by embedding-enhanced texttiling. In *INTERSPEECH'16*, 2016.

[31] A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and B. Dolan. A neural network approach to context-sensitive generation of conversational responses. In *NAACL'15*, pages 196–205, 2015.

[32] H. Sugiyama, T. Meguro, R. Higashinaka, and Y. Minami. Open-domain utterance generation for conversational dialogue systems using web-scale dependency structures. In *Proc. SIGDIAL*, pages 334–338, 2013.

[33] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[34] M. A. Walker, R. Passonneau, and J. E. Boland. Quantitative and qualitative evaluation of darpa communicator spoken dialogue systems. In *ACL '01*, pages 515–522.

[35] R. S. Wallace. *The anatomy of ALICE*. Springer, 2009.

[36] C. Wang, Y. Song, A. El-Kishky, D. Roth, M. Zhang, and J. Han. Incorporating world knowledge to document clustering via heterogeneous information networks. In *KDD*, pages 1215–1224, 2015.

[37] C. Wang, Y. Song, H. Li, M. Zhang, and J. Han. Knowsim: A document similarity measure on structured heterogeneous information networks. In *ICDM*, pages 1015–1020, 2015.

[38] C. Wang, Y. Song, H. Li, M. Zhang, and J. Han. Text classification with heterogeneous information network kernels. In *AAAI*, pages 2130–2136, 2016.

[39] H. Wang, Z. Lu, H. Li, and E. Chen. A dataset for research on short-text conversations. In *EMNLP'13*, pages 935–945, 2013.

[40] J. Williams, A. Raux, D. Ramachandran, and A. Black. The dialog state tracking challenge. In *SIGDIAL'13*, pages 404–413, 2013.

[41] R. Yan, L. Kong, C. Huang, X. Wan, X. Li, and Y. Zhang. Timeline generation through evolutionary trans-temporal summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 433–443, 2011.

[42] R. Yan, M. Lapata, and X. Li. Tweet recommendation with graph co-ranking. In *ACL '12*, pages 516–525.

[43] R. Yan, Y. Song, and H. Wu. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 55–64, 2016.

[44] R. Yan, X. Wan, J. Otterbacher, L. Kong, X. Li, and Y. Zhang. Evolutionary timeline summarization: A balanced optimization framework via iterative substitution. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 745–754, 2011.

[45] K. Yao, G. Zweig, and B. Peng. Attention with intention for a neural network conversation model. *arXiv preprint arXiv:1510.08565*, 2015.

[46] K. Zhai and D. J. Williams. Discovering latent structure in task-oriented dialogues. In *ACL'14*, pages 36–46, 2014.

[47] B. Zhang, J. Su, D. Xiong, Y. Lu, H. Duan, and J. Yao. Shallow convolutional neural network for implicit discourse relation recognition. In *EMNLP*, pages 2230–2235, 2015.

[48] X. Zhou, D. Dong, H. Wu, S. Zhao, R. Yan, D. Yu, X. Liu, and H. Tian. Multi-view response selection for human-computer conversation. In *EMNLP'16*, 2016.