information

MDPI

Article

MOLI: Smart Conversation Agent for Mobile Customer Service

Guoguang Zhao [1], Jianyu Zhao [1], Yang Li [1], Christoph Alt [2], Robert Schwarzenberg [2], Leonhard Hennig [2], Stefan Schaffer [2], Sven Schmeier [2], Changjian Hu [1] and Feiyu Xu [1,*]

[1] No. 10, East Xibeiwang Rd., Haidian District, Beijing 100094, China; zhaogg1@lenovo.com (G.Z.), zhaojy10@lenovo.com (J.Z.); liyangkeda@126.com (Y.L.); hucj1@lenovo.com (C.H.)

[2] Alt-Moabit 91c, 10559 Berlin, Germany; christoph.alt@dfki.de (C.A.); robert.schwarzenberg@dfki.de (R.S.); leonhard.hennig@dfki.de (L.H.); stefan.schaffer@dfki.de (S.S.); schmeier@dfki.de (S.S.)

* Correspondence: fxu@lenovo.com

Received: 8 January 2019; Accepted: 12 February 2019; Published: 15 February 2019

Abstract: Human agents in technical customer support provide users with instructional answers to solve a task that would otherwise require a lot of time, money, energy, physical costs. Developing a dialogue system in this domain is challenging due to the broad variety of user questions. Moreover, user questions are noisy (for example, spelling mistakes), redundant and have various natural language expressions. In this work, we introduce a conversational system, MOLI (the name of our dialogue system), to solve customer questions by providing instructional answers from a knowledge base. Our approach combines models for question type and intent category classification with slot filling and a back-end knowledge base for filtering and ranking answers, and uses a dialog framework to actively query the user for missing information. For answer-ranking we find that sequential matching networks and neural multi-perspective sentence similarity networks clearly outperform baseline models, achieving a 43% error reduction. The end-to-end P@1(Precision at top 1) of MOLI was 0.69 and the customers' satisfaction was 0.73.

Keywords: question and answer; dialog systems; semantic matching

1. Introduction

For many companies, customers can seek customer support from multiple channels such as web page, Facebook or APP. Besides, according to the research of China Information Industry Network (CNII), customer service and support is a sizable and growing market globally, as well as in China. In response to tremendous demand in our company and market we develop our smart customer service MOLI for mobile.

"My Wi-Fi is not working anymore!"—most mobile device users probably have faced this or a similar questions in the past. Solving such questions is the task of customer support agents (CSAs). For frequent questions and user intents, for which solutions often exist in the form of user guides and question-answer knowledge base (QA-KB), this is a repetitive and time consuming process. Automating such conversations would significantly reduce the time CSAs have to invest in solving common questions, which they could then spend on more complex or previously unseen customer problems [1].

Recent advances in dialog systems have led to successful applications in domains such as restaurant [2] and flight bookings [3], providing a convenient way for users to interact with backend services and knowledge bases in natural language, via speech or text-based input. Developing a dialog system for technical customer support presents additional challenges due to the broad variety of topics and tasks that need to be handled. The task is made even more challenging by the fact that the dialogs

are often noisy, contain grammatical errors, and incomplete user turns. They also refer to concepts and entities not recognized by standard NER tools (e.g., devices, components). Due to the non-technical background of most customers, problem descriptions can be ambiguous, too colloquial with respect to the more formalized, technical QA knowledge base texts, and possibly miss important information that is necessary to identify a unique and correct solution. CSAs therefore often query customers for contextual information, such as device model and mobile carrier, in order to identify the exact issue and to select a good answer.

With the work described in this paper, we aim to automatize this task of matching instructional answers from a QA-KB to user queries described by users in online support chats. We describe an approach to conversational question answering in the little-explored domain of technical customer support. Our approach selects the best answer from a QA-KB in a dialog-oriented fashion, using intent classification to narrow down answer candidates and in particular to pro-actively query the user for missing information.

2. Related Work

Existing work on dialog systems in the customer service domain has focused on dialog modeling [4] for answering Ubuntu OS-related questions, and single-turn question answering in the insurance domain [5]. Both studies show that it is in principle possible to handle longer conversations in an unsupervised fashion and answer complex questions with the help of a noisy training set and an unstructured knowledge source. Lowe et al. [4] use a large corpus of support conversations in the operating system domain to train an end-to-end dialog system for answering customer questions. Their results suggest that end-to-end trained systems can achieve good performance but perform poorly on dialogs that require specific domain knowledge which the model possibly never observed.

In contrast, in our work we adopt a classical classification approach followed by semantically matching a user question to a set of results from a QA-KB, in order to cope with the limited amount of training data. The work of Feng et al. [5] focuses on answer matching and selection for a spoken question answering system. The authors show that a standard CNN (Convolutional Neural Network)-based approach with a large number of filters can achieve good performance on this task, but it is limited to single-turn question-response conversations.

Most previous works on sentence similarity modeling focus on feature engineering. Several types of features have been proved useful, including: (1) string-based, including n-gram features [6] and features used in machine translation evaluation [7]; (2) knowledge-based, using lexical resources (e.x. WordNet) in Fern and Stevenson' work [8]; (3) syntax-based, such as, modeling divergence of dependency relation between two sentences in Das and Smith' work [9]; (4) corpus-based, using distributional models such as using latent semantic analysis to catch the features [10]. Recent work has been changed from hand-crafted features to modeling with distributed representations under neural network architectures. Collobert and Weston [11] used convolutional neural networks to be trained jointly for multiple tasks in NLP (Natural Language Processing) with the same shared weights. Kalchbrenner et al. [12] utilized a dynamic k-max pooling to better model inputs of varying sizes in a convolutional neural network to model sentences. Kim [13] proposed several improvements to the convolutional architecture of Collobert and Weston, including the use of fixed word vectors and varying sizes for convolution windows. Hu et al. [14] used convolutional neural net works to combine sentence modeling in a layer-by-layer composition method. A variety of other models have been proposed for similarity tasks (Weston et al. [15], Huang et al. [16], Andrew et al. [17]). Tai et al. [18] proposed a tree-based LSTM (Long Short-Term Memory).

In our approach we follow the architecture proposed by He et al. [19] to handle the ambiguity and variability of linguistic expression when modeling sentence semantic similarity. Their work proposes a multi-perspective model for comparing sentences. The authors first use a CNN model to extract features at multiple levels of granularity and then compute multiple similarity metrics to measure sentence similarity. Wu et al. [20] propose a sequential matching network (SMN) which matches

two sentences in the context on multiple granularity, and distills important matching information from each pair with convolution and pooling operations followed by a recurrent network to model sentence relationships.

In contrast to standard QA tasks such as extractive QA [21] and opensequential matching network-domain answer selection [22], the task we address focuses on identifying instructional answers that match the user's question and its context, and hence cannot simply be gleaned from a text corpus. Although question answering and dialog systems recently received a lot of attention, work combining both is still sparse.

3. Conversational Question Answering

3.1. Problem Formalization

The goal of our approach was to identify the answer a_i from a corpus of N QA pairs $QA = \{(q_1, a_1), \ldots, (q_n, a_n)\}$ that best matched the user's question as expressed by a sequence of user turns $T = t_1, \ldots, t_T$. q_i is a representative question such as "How to setup email" which prototypically stands for other questions that can be answered by a_i. As Figure 1 illustrates, QA pairs as well as user turns are expressed as free text.

For each pair (q_i, a_i), there exists meta data, recorded in the form of properties $p_{i,1}, \ldots, p_{i,m}$. Properties describe the context to which a given QA pair applies, such as a specific device name and operating system version. They typically take a single value from a limited set of possible values.

At each turn t_k during the conversation our system estimates the utility $u_{i,k} \in \mathbb{R}_{\geq 0}$ of answer a_i given the current context $C =< t_1, \ldots, t_k, P_k >$ where P_k is a list of relevant properties that have already been identified (filled 'slots').

Moli: Hello there. I'm Moli, your virtual agent. How may I help you today?
Customer: Hi, i am a new moto user. How do I add a google account? Can you tell me how to do it.
Moli: What's your phone model (for example "Moto G5 Plus" or "Moto Z2 Play")?
Customer: moto g3
Moli: You can use multiple Google Accounts and Microsoft Exchange Active Sync accounts on your device. Add account: You may need to obtain details from system administrator about the service to which the account connects.
For example, you may need to know the account's domain or server address.
1. Touch Apps>settings
2.Touch Accounts
3.Select the type of account, e.g.Exchange,Email,Google

Figure 1. Example chat about an "email account setup".

3.2. Dialog System Overview

Figure 2 gives an overview of the overall architecture of the dialog system. Customers can interact with the system either by entering free text or by selecting one of a set of predefined choices suggested by the system. Choices can be of different types—frequently asked questions at the start of a conversation, or, for instance, product names if the system queried for the customer's device in the previous turn. Upon receiving a user turn t_k, the natural language understanding (NLU) component's task was to transform t_k into a structured representation. It first performed sentence segmentation and tokenization of the input using the Stanford CoreNLP toolkit [23], and then determined the question type, namely, how-to or others. In addition, the NLU component performed intent classification to identify user intent that represents a similar question set. The intent was used for narrowing down candidate standard QA pairs from QA-KB. Then the slot filling (SF) component is used for identifying entities such as product names and attributes, which are linked to concepts in a product knowledge graph (KG). SF is based on a combination of template and sequence classification approaches. After question type classification,intent classification and slot filling, the system determined the truly concerned specific question of user. Since SF component is based on

combination of template and sequence classification approaches, the model with high accuracy but low recall. We add a semantic matching component to improve recall without decreasing accuracy.

The main task of the dialog manager (DM) was to maintain the dialogue context (semantic frame), which encoded all available structured information, and decided on the next system action. If the semantic frame was not complete, that is, it was lacking a slot value such as product information, the DM can re-ask, confirm, or clarify to update the dialogue context and keep the semantic frame unambiguous and complete. If a product name was not detected, the DM may also query the customer database for the most recent product purchased by this customer. Given the question type, the intent category and the values of already filled slots, the DM retrieved the list of potential QA pairs by querying the QA-KB. It then either asked the customer for more information to fill empty slots (which are specified by the properties of the QA pairs) to narrow down this list, or it runs the semantic matching component to rank the remaining QA pairs The DM can also ask for confirmation or disambiguation of a slot filler extracted by the NLU component. DM action choices were passed through a template text-based natural language generation component (NLG) or were converted into option choices represented by buttons in the user interface. At any time during the conversation, the system or the user may choose to refer the conversation to a human support agent.

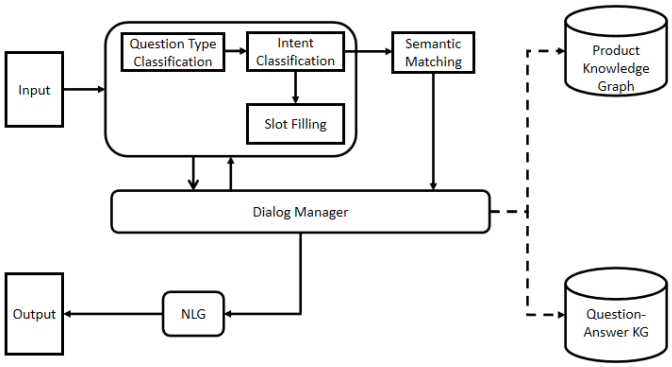

Figure 2. Dialog system architecture.

4. The Proposed Model

4.1. Question Type Classification

Question type classification is the entrance of NLU, which is important for the performance of the whole system. Here, we define n-gram information as a local semantic feature and long dependency relationship between words or phrases as a global structure feature. Most existing question type classification models either learn little structure information or just rely on pre-defined structures, leading to degradation of performance and generalization capability. To address this issue, we propose a sandwich neural network (SNN) to learn semantic and structure representations automatically.

SNN contains four parts: first LSTM layer, CNN and pooling layer, second LSTM layer and concatenation and loss layer. The first LSTM layer was inspired by DSCNN [24] to adjust the word representation which takes the context into account. CNN and pooling was used to learn local n-gram semantic representation. The second LSTM layer was inspired by C-LSTM [25] to use the filter maps after convolution to represent the high-level phrase representation and feed it into following LSTM to learn long-dependency structure representation. The last concatenation and loss layer was used to concatenate these two representations as a new one, to compute loss through corss-entorpy. Figure 3 shows the architecture of our model, where CNN is in the middle of two LSTM layers like a sandwich. Then, we will describe SNN in detail.

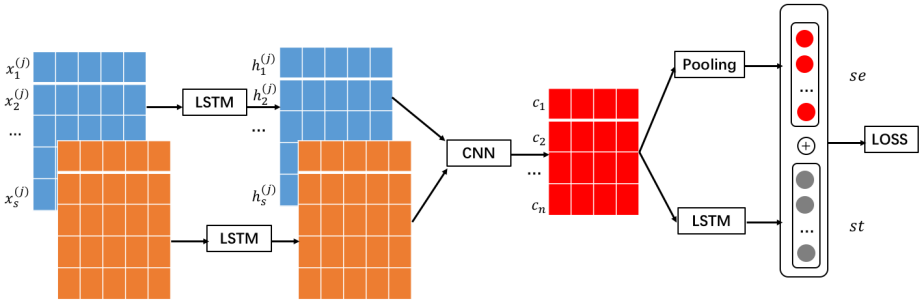

Figure 3. Dialog system architecture.

4.1.1. First LSTM Layer

We let our model's input be a sentence of length s : $[w_1, w_2, ..., w_s]$, c be the whole number of word embedding versions and $\boldsymbol{x}_i^{(j)}$ is the i^{th} word's embeddings of the j^{th} version. The common word embedding versions are Word2vec and Glove.

Our model's first layer consisted of LSTM networks which processed different versions of word embeddings. For every version of word embeddings, there was an according LSTM network where the input $\boldsymbol{x}_t \in \mathbb{R}^d$ is the d-dimensional word embedding for w_t. The LSTM layer will produce a hidden state representation $\boldsymbol{h}_t \in \mathbb{R}^d$ at each time step. The hidden state representations will be set as the output of LSTM layers:

$$\boldsymbol{h}^{(i)} = [\boldsymbol{h}_1^{(i)}, \boldsymbol{h}_2^{(i)}, ..., \boldsymbol{h}_t^{(i)}, ..., \boldsymbol{h}_s^{(i)}] \tag{1}$$

for $i = 1, 2, ..., c$.

4.1.2. CNN Layer

The second layer was a CNN. To utilize multiple kinds of word embeddings, we applied a filter $\boldsymbol{F} \in \mathbb{R}^{c \times d \times l}$, where l is the size of convolution window. The i^{th} version of word embedding produce the hidden state sequence $\boldsymbol{h}^{(i)}$, which forms one channel of the feature map. Then these feature maps are stacked c-channel feature maps $\boldsymbol{X} \in \mathbb{R}^{c \times d \times s}$.

Afterwards, filter F convolved with the window vectors (l-gram) at each position to generate a feature map $\mathbf{c} \in \mathbb{R}^{s-l+1}$; c_k is the element of the feature map $\mathbf{c}$ for window vector $\boldsymbol{X}_{k:k+l-1}$ at position k and it is produced as follows:

$$c_k = f(\sum_{i,j,r} (\boldsymbol{F} \bigodot \boldsymbol{X}_{k:k+l-1})_{i,j,r}), \tag{2}$$

where $\bigodot$ denotes element-wise multiplication.

The n feature maps generated from n filters can be rearranged through column vector concatenation method to form a new representation,

$$\mathbf{W} = [\mathbf{c}_1; \mathbf{c}_2; ...; \mathbf{c}_n] \tag{3}$$

Each row $\mathbf{W}_j$ of $\mathbf{W} \in \mathbb{R}^{(s-l+1) \times n}$ is the feature map generated from n filters for the window vector at position j. The new successive higher-level representations were then fed into the last LSTM layer.

Here, a max-over-time pooling layer was added after the convolution neural network. The pooling result of the feature map $\mathbf{c}$ is :

$$p = max(c_1, c_2, ..., c_{s-l+1}) \tag{4}$$

These pooling results are used as our local semantic representation $\boldsymbol{se} \in \mathbb{R}^n$:

$$\boldsymbol{se} = [p_1, p_2, ..., p_n] \tag{5}$$

4.1.3. Second LSTM Layer

We used the same number of filters n to denote the dimension in this LSTM layer for easy and fair fusion in the latter, and use the last hidden unit of LSTM as global structure representation $\boldsymbol{st} \in \mathbb{R}^n$.

4.1.4. Concatenation and Loss Layer

Thus, we got the local semantic representation $\boldsymbol{se}$ and global structure representation $\boldsymbol{st}$. Then we concatenated $\boldsymbol{se}$ and $\boldsymbol{st}$ to get the sentence representation and compute loss through cross entropy.

4.2. Intent Category Classification

The correct intent can reduce the number of candidate QA pairs significantly. Currently, the data set contains 60 intents such as "Bluetooth", "Screen Unlock", "Google Account", etc.

The intent category classifier estimates the probability $p(I|t_k)$ where I represents intent. Our baseline approaches are GBDT (Gradient Boosting Decision Tree) and a linear SVM (Support Vector Machine). For feature extraction, at first the t_k was tokenized, followed by stop-word removal and transformation into a bag-of-words representation. The features were term frequency-reverse document frequency (TF-IDF) weighted unigram and bigram features. We also implemented a bidirectional LSTM model(BiLSTM). In this model, each $w_i \in t_k$ was represented by an embedding $e_i \in R^d$ that we obtain from pre-trained distributed word representations $E = [e_1, \ldots, e_W]$. The BiLSTM output was passed to a fully-connected layer followed by a ReLU (Rectified Linear Unit) non-linearity and softmax normalization, s.t. $p(I|t_k)$ was computed as follows:

$$SM(ReLU(FC(BiLSTM(E)))(t_k) \tag{6}$$

4.3. Semantic Matching

We assume that the QA pair with the highest semantic similarity to the question expressed in turn t_k (and previous turns) will be of the highest utility to the user. After question type and intent category classification we obtain an initial set of candidates, QA_{init}, by retrieving all QA pairs from the knowledge base that are relevant to the question type and intent category. Following a common information retrieval approach, we then used a pairwise scoring function $S(q_i, a_i, C)$ to sort QA_{init} by utility, where $(q_i, a_i) \in QA_{init}$.

TF-IDF

TF-IDF means term frequency-inverse document frequency. Our first baseline computes S with a TF-IDF weighted bag-of-words representation of q_i, a_i and t_k to estimate the semantic relatedness by cosine similarity $cos(v_i, v_k)$ between the feature vectors of the QA pair, v_i, and the user turn, v_k.

WMD

The second baseline leverages the semantic information of distributed word representations [26]. To this end, we replace the tokens in q_i, a_i and t_k with their respective embeddings and then compute the word mover distance [27] between the embeddings.

SMN

In addition to the baselines we use a sequential matching network (SMN) [20], which treats semantic matching as a classification task. The SMN first represents q_i, a_i and t_k by their respective sequence of word embeddings E_i and E_k before encoding both separately with a recurrent network, a gated recurrent unit (GRU) [28] in this case. A word-word similarity matrix M_w and a sequence-sequence similarity matrix M_s is constructed from E_i and E_k, and important matching information is distilled into a matching vector v_m via a convolutional layer followed by max-pooling. v_m is further projected using a fully connected layer followed by a softmax.

MPCNN

In this section, we present innovative solutions that incorporate multi-info and context information of user questions into multi-perspective CNN(MPCNN) to fulfill question paraphrase identification. The architecture of model is shown in Figure 4. Our model has two same subnetworks that processing t_k and $q_i a_i$ in parallel after getting context by GRU.

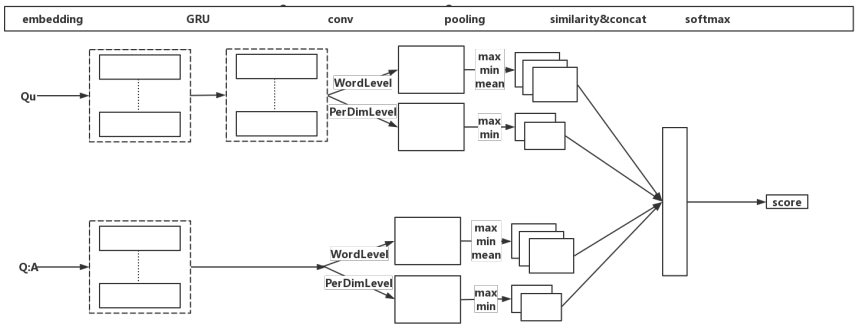

Figure 4. Multi-perspective sentence similarity network with gated recurrent unit (GRU).

(1) Multi-info

To the data, t_k is quite long but q_i in our QA-KB is short and contains less information. Besides, the a_i is quite long and contains some information that related to t_k. In this work, we concat q_i and a_i of QA-KB then to compute S($q_i a_i$, t_k). User queries are always concerned with a specific product but some related standard questions for different products may be the same in the QA-KB. For example in Figure 1 "moto g3" is a mobile name. For a same question, if the product of the question is different, it will influence the matching result. We replace these specific mobiles by the same word "Mobile" directly. In this paper, we use the product-KB and CRF (Conditional Random Field) algorithm to recognize the mobile from t_k. The left part of Figure 5 indicates the structure of the product-KB. In product-KB, every mobile has its surface names which are mined from the chat log.

Product-KB hardly contains all mobiles and their surface names so we use CRF to recognize the mobile names from the input user question as a supplement. There are two level features used in CRF, char level ngrames and word level ngrams. The maximum char level ngram is 6 and word level ngram is 3. By using the multi-info of product-KB and answer information, the precision of semantic matching is improved.

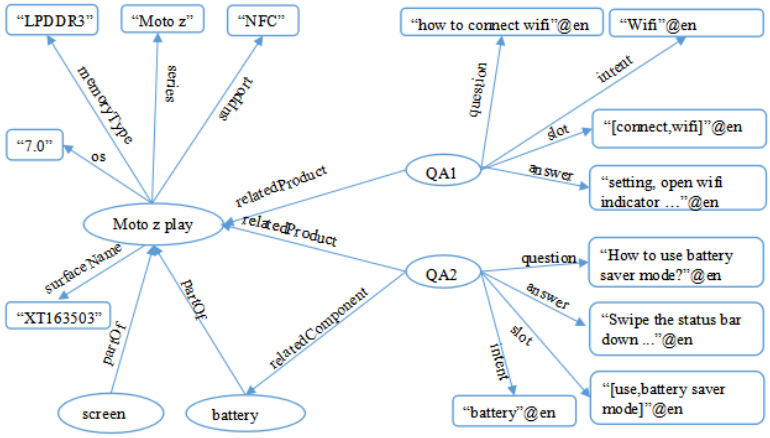

Figure 5. Structure of the product-knowledge base (KB) and question-answer (QA)-KB.

(2) Context Multi-Perspective CNN

After getting the multi-info, the input of our neural model are t_k and $q_i a_i$. Given a user query t_k and a response candidate $q_i a_i$, the model looks up an embedding table and represents t_k and $q_i a_i$ as $t_k = [e_{u,1}, e_{u,2}, \ldots, e_{u,L}]$ and $q_i a_i = [e_{s,1}, e_{s,2}, \ldots, e_{s,L}]$ respectively, where $e_{u,j}$ and $e_{s,j} \in R^d$ are the embeddings of the j-th word of t_k and $q_i a_i$ respectively. L is the max length of two input sequences. Before feed into multi-perspective CNN, t_k is transformed to hidden vectors conM_{Qu} by GRU. Suppose that $\text{conM}_{t_k} = [h_{u,1}, h_{u,1}, \ldots, h_{u,L}]$ are the hidden vectors of t_k, then $h_{u,i}$ is defined by

$$z_i = \sigma(W_z e_{u,i} + U_z h_{u,i-1}) \tag{7}$$

$$r_i = \sigma(W_r e_{u,i} + U_r h_{u,i-1}) \tag{8}$$

$$\overline{h}_{u,i} = tanh(W_h e_{u,i} + U_h(r_i \odot h_{u,i-1})) \tag{9}$$

$$h_{u,i} = z_i \odot \overline{h}_{u,i} + (1 - z_i) \odot h_{u,i-1}, \tag{10}$$

where $h_{u,0} = 0$, z_i and r_i are an update gate and a reset gate respectively, $\sigma(.)$ is a sigmoid function, and W_z, W_r, W_h, U_z, U_r, U_h are parameters.

Because $q_i a_i$ is not a sequential sentence the model only gets context information of t_k and learns long-term dependencies by GRU. conM_{t_k} and $q_i a_i$ are then processed by the same neural networks. The paper applies to both word level convolutional filters and embedding level convolutional filters. Word level filters operate over sliding windows while considering the full dimensionality of the word embeddings, like typical temporal convolutional filters. The embedding level filters focus on information at a finer granularity and operate over sliding windows of each dimension of the word embeddings. Embedding level filters can find and extract information from individual dimensions, while word level filters can discover broader patterns of contextual information. Both kinds of filters are allowed to extract more information for richer our model.

For every output vector of convolutional filter, the model converts it to a scalar by pooling layer. Pooling helps a convolutional model retain the most prominent and prevalent features, which is helpful for robustness our model. Max pooling is a widely used pooling layer, which applies max operation over the input vector and returns the maximum value. In addition to using max pooling, our model also uses min pooling and mean pooling.

4.4. Dialogue Manager

For the conversation, we adopted the method based on finite state machine (FSM) to manage it. We set up eight intermediate states for conversation besides "start" and "close" states, such as: "Init", "SlotFull", "SlotNotFull", "SlotClarify", "IntentVerify", "DeliverAnswer", "WaitUserInput" and "ErrorHandling" as shown in the Figure 6. We explain the meaning of each state separately as shown in Table 1. Besides, in order to clearly see the jump logic between states, we use arrow lines to indicate the next state to jump , as shown in Figure 6.

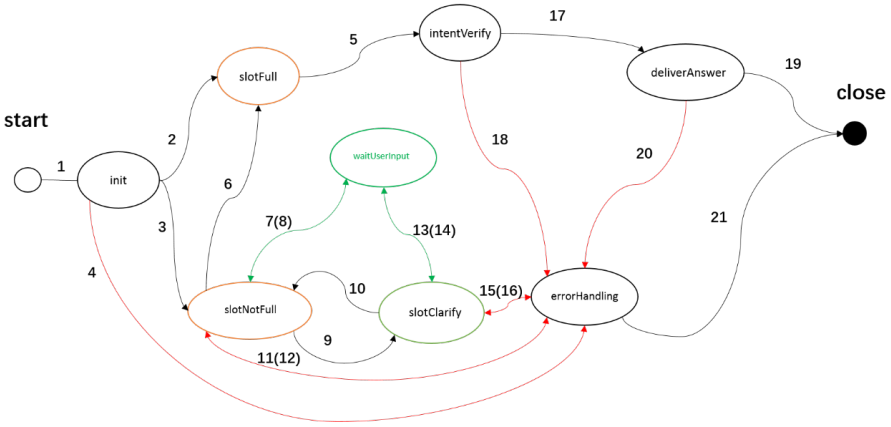

Figure 6. Finite state machine for task-oriented dialog.

Table 1. Definition of states.

No.	State Name	Definition
1	Start	Refers to "start" state of the state machine.
2	Init	It means initialization state.
3	Slotfull	It means all necessary slots are filled and verified completely.
4	SlotNotFull	It means not all necessary slots are filled or verified completely.
5	SlotClarify	It means that some necessary slots need verify by user or knowledge base.
6	IntentVerify	It means some intent from users need verify.
7	DeliverAnswer	It means that in current state our FSM is delivering answer to our users.
8	WaitUserInput	In this state our FSM accepts users' input.
9	ErrorHandling	The state is used for dealing with errors happening in the state jumping process.
10	End	It refers to terminate the task.

5. Experiments and Discussion

In this section, we evaluate our approaches for question type classification, question intent category classification, semantic matching as well as end-to-end performance of MOLI. Next, we will introduce the dataset, QA-KB, product-KB and experiment results separately.

5.1. Data Set, QA-KB and Product-KB

The chat transcript data set mainly consists of first contact transcripts, in which the customer's question or intent is explicitly stated. Each transcript includes the full text of the chat, speaker ids for each message, a product id, optionally question type and an intent category assigned by the customer service agent.

From this corpus, we extracted a dataset of 80,216 user turns, which are manually labeled with question type information by the CSAs. Table 2 shows the distribution over question types contained in the dataset. Out of the 30,593 how-to questions, 6808 have an intent category assigned, for which the distribution over the top 30 categories (out of 60) is shown in Figure 7.

Table 2. Question type statistics.

Question Type	Num	Percentage(%)
How To	30,953	38.6
Other	49,263	61.4

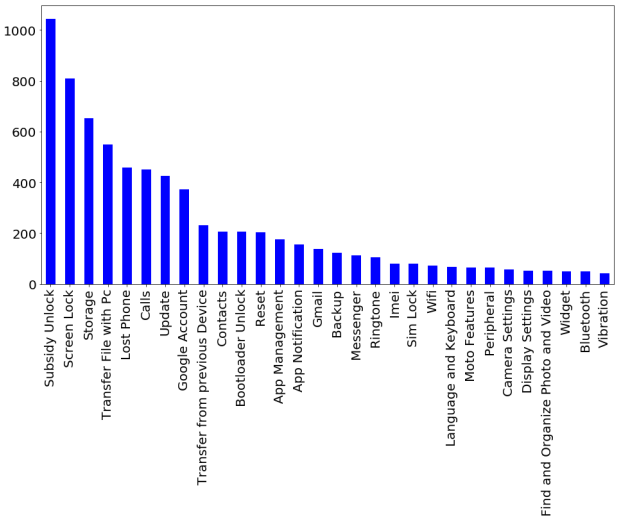

Figure 7. Distribution of intent categories (top 30) for user question.

The KB stores the QA pairs and its relevant products. Figure 5 indicates the structure of our KB. The left part is the parameters of mobile product and the right part is QA pairs. In the current version, KB includes 20 mobile products, 242 standard QA pairs. Our KB in total includes more than 150,000 triples.

5.2. Question Type Classification

In this section, we firstly split the data set into 80/20 training and test sets, respectively. In the paper, we use 300 dimensional GloVe word embeddings [29]. Hyper-parameter selection was done on the training set via five-fold cross validation and results averaged over multiple runs are reported on the test set and Table 3 shows the evaluation results. As we see, SNN outperforms the baseline models prominently. For example, in the sentence "I want to get support on the steps of factory model setting", the structure "support on ... steps" contributes a lot to the classification. Any part of the structure ("support" or "steps") may make mistakes.

Table 3. Question type classification results.

Model	Precision	Recall	F1
C-LSTM	0.92	0.90	0.91
DSCNN	0.93	0.91	0.92
SNN	0.96	0.94	0.95

5.3. Intent Category Classification

In this section, we also firstly split the dataset into 80/20 train and test sets, respectively. Hyper-parameter selection is done on the training set via 5-fold cross validation and results averaged over multiple runs are reported on the test set. For baseline model BiLSTM we use 300 dimensional GloVe word embeddings [29]. Table 4 shows the evaluation results on the dataset. The baseline model SVM, even outperforming the BiLSTM model.

Table 4. Intent category classification results for user question.

Model	Precision	Recall	F1
GBDT	0.67	0.68	0.67
BiLSTM	0.68	0.70	0.69
SVM	0.74	0.76	0.75

From the specific every category results of SVM in Table 5 we find that some categories (e.g., “Google account and transfer from previous device”) achieve a disproportional lower performance. For example, “Google account” is often confused with "reset as a Google account" is generally a main topic when trying to reset a device (e.g., “Android smartphone”). It is also noteworthy that “subsidy unlock”, “bootloader unlock” and “screen lock” are frequently confused. This is best illustrated by the example “Hi i need pin for unlock red to my moto g”, which has the true category “Subsidy Unlock” but is categorized as “screen lock”. Without knowledge about the mobile phone and contracts domain it is very difficult to understand that the customer is referring to a “pin” (subsidy unlock code) for “red” (mobile service provider) and not the actual PIN code for unlocking the phone. This example also symbolizes a common problem in smart customer support, where users unfamiliar with the domain are not able to describe their information need in the domain-specific terminology.

Table 5. Intent category classification results for user question, top 10 categories.

Model	Precision	Recall	F1
Subsidy unlock	0.83	0.93	0.88
Screen lock	0.84	0.92	0.88
Storage	0.79	0.85	0.82
Transfer file w. PC	0.77	0.91	0.83
Lost phone	0.87	0.91	0.89
Calls	0.79	0.87	0.83
Google account	0.64	0.72	0.68
Update	0.77	0.92	0.84
Bootloader unlock	0.93	0.67	0.78
Transfer p. device	0.67	0.74	0.70

5.4. Semantic Matching

For all models except TF-IDF, we use 300 dimensional GloVe word embeddings [29]. To obtain negative samples, for each t_k, we randomly selected five standard queries with the same intent and five standard queries with different intents. To alleviate the impact of unbalanced training data, we oversampled positive samples. As the standard questions q_i of most QA pairs (q_i, a_i) are usually less than 10 tokens, we also evaluate the impact on model performance when adding the answer a_i as additional context (up to 500 characters) to q_i.

Table 6 shows the P@1 of each model on our data. We see that the MPCNN and MPCNN_GRU (MPCNN with a Gated Recurrent Unit) outperform the unsupervised baseline approaches, with a 43% error reduction achieved with the MPCNN_GRU model. Intuitively it makes sense to provide the models with additional context that can be used to learn a better representation of semantic similarity. The SMN’s P@1 are much lower than MPCNN models, even only slightly higher than these unsupervised models.

Table 6. Semantic matching results for user question.

Model	Without Answer	With Answer
TF-IDF	0.62	0.60
WMD	0.60	0.58
SMN	0.62	0.68
MPCNN	0.72	0.84
MPCNN_GRU	0.72	0.85

5.5. The Importance of Intent Classification for Semantic Matching

Question intent classification is an important step to narrow down candidate answers. In this section, we compare with baseline models to highlight the effectiveness of intent classification. The baseline models used the same network as MPCNN and MPCNN_GRU, without intent

classification so the models were matching with all QA pairs directly. Table 7 shows that the precision of semantic matching with intent outperformed baseline models.

Table 7. Semantic matching results on baseline for user question.

	Without Intent	With Intent
MPCNN	0.63	0.84
MPCNN_GRU	0.65	0.85

5.6. End-to-End Performance of MOLI

In this section, we display the end-to-end performance of MOLI and compare MOLI with baseline system to highlight the effectiveness of related component. In detail, we list the baseline performance, and then we list the performance with question type classification, intent category recognition, semantic matching improved respectively in Sections 5.1–5.3. The detailed methods are SNN, SVM, MPCNN_GRU respectively, so we name the baseline models baseline_QT_{SNN}, baseline_IC_{SVM}, baseline_SM_{MPCNN_GRU}. At last, we show the performance with all the above components improved together. Besides, in order to prove the effectiveness of semantic matching, we designed the MOLI-SM model, which removed the semantic matching component based on the MOLI. Table 8 shows the P@1 and feedback score of each system. The feedback score is calculated by user's action. At the end of a session in the system, there is a feedback mechanism where you can grade the recommend answer. There were five level scores that the user could choose. If the score was four or five then we think the answer was useful for the user. In the table, the results show that our improvements were useful.

Table 8. End-to-end performance.

	P@1	feedback Score
baseline	0.45	0.49
baseline_QT_{SNN}	0.48	0.52
baseline_IC_{SVM}	0.50	0.55
baseline_SM_{MPCNN_GRU}	0.51	0.56
MOLI-SM	0.64	0.67
MOLI	0.69	0.73

6. Conclusions

In this paper, we describe our smart customer system MOLI in detail with many innovative NLP techniques. We presented a first approach for conversational question answering in the complex and little-explored domain of technical customer support. Our approach matches a user's question with the most relevant answer from a knowledge base. It does so in a conversational manner, by asking for, and clarifying required information if necessary. Our approach incorporates several separate models to determine an answer. Most notably, it performs question type and intent classification for a dataset with 60 intent categories, slot filling, and semantic answer matching. We observe that while supervised models, both neural and standard ones such as decision trees and SVMs, perform reasonably well on the individual tasks, there is still room for improvement. As many previous authors have shown in other domains, such models can benefit from joint training and end-to-end task modeling.

Our experiments were conducted with a dataset of noisy, real-world chat transcripts, which we plan to make available to the community in the near future. Future research directions include end-to-end, joint modeling of the question type and intent classification, slot filling and semantic matching subtasks, as well as updating the dialogue manager to account for nested, non-linear conversations, and maintaining multiple dialog hypotheses.

Author Contributions: G.Z., J.Z. and Y.L. write the manuscript and help to revise it; C.A., R.S., L.H., S.S. (Stefan Schaffer) and S.S. (Sven Schmeier) help to revise the manuscript; C.H. and F.X. Offer funding for ths paper. Thanks for all authors' contribution.

Funding: The funding is from Lenovo Research.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Li, Y.; Miao, Q. Geng, J. Question Answering for Technical Customer Support. In *7th CCF International Conference*; Springer: Cham, Switzerland, 2018; pp. 3–15.
2. Wen, T.H.; Vandyke, D. Mrksic, N. A Network-based End-to-End Trainable Task-oriented Dialogue System. *arXiv* **2008**, arXiv:1604.04562.
3. Kahaduwa, H.; Pathirana, D.; Arachchi, P.L. Question Answering system for the travel domain. In Proceedings of the Moratuwa Engineering Research Conference (MERCon), Moratuwa, Sri Lanka, 29–31 May 2017; pp. 449–454.
4. Lowe, R.T.; Pow, N.; Serban, I.V. Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue Discourse* **2017**, *8*, 31–65.
5. Feng, M.; Xiang, B.; Glass, M.R. Applying deep learning to answer selection: A study and an open task. In *Proceedings of the 3rd International Proceedings on Automatic Speech Recognition and Understanding (ASRU)*; IEEE: Piscataway NJ, USA, 2015; pp. 813–820.
6. Wan, S.; Dras, M.; Dale, R.; Paris, C. Using Dependency-based Features to Take the Para-farce out of Paraphrase. In Proceedings of the Australasian Language Technology Workshop, Sydney, Australia, 30 November–1 December 2006; pp. 131–138.
7. Madnani, N.; Tetreault, J.; Chodorow, M. Re-examining machine translation metrics for paraphrase identification. In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Montreal, QC, Canada, 3–8 June 2012; pp. 182–190.
8. Fernando, S.; Stevenson, M. A Semantic Similarity Approach to Paraphrase Detection. Availiable online: https://pdfs.semanticscholar.org/d020/eb83f03a9f9c97e728355c4a9010fa65d8ef.pdf (accessed on 14 February 2019).
9. Das, D.; Smith, N.A. Paraphrase identification as probabilistic quasi-synchronous recognition. In Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, Suntec, Singapore, 2–7 August 2009; pp. 468–476.
10. Weiwei Guo and Mona Diab. Modeling sentences in the latent space. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Jeju Island, Korea, 8–14 July 2012; pp. 864–872.
11. Collobert, R.; Weston, J. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 5–9 July 2008; pp. 160–167.
12. Kalchbrenner, N.; Grefenstette, E.; Blunsom, P. A convolutional neural network for modelling sentences. *arXiv* **2014**, arXiv:1404.2188.
13. Kim, Y. Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods for Natural Language Processing, Doha, Qatar, 25–29 October 2014; pp. 1746–1751.
14. Hu, B.; Lu, Z.; Li, H.; Chen, Q. Convolutional neural network architectures for matching natural language sentences. *arXiv* **2015**, arXiv:1503.03244.
15. Weston, J.; Bengio, S.; Usunier, N. Wsabie: Scaling up to large vocabulary image annotation. In Proceedings of the International Joint Conference on Artificial Intelligence, Barcelona, Spain, 16–22 July 2011; pp. 2764–2770.
16. Huang, P.S.; He, X.; Gao, J.; Deng, L.; Acero, A.; Heck, L. Learning deep structured semantic models for web search using clickthrough data. In Proceedings of the 22nd ACM International Conference on Information Knowledge Management, San Francisco, CA, USA, 27 October–1 November 2013; pp. 2333–2338.
17. Andrew, G.; Arora, R.; Bilmes, J.; Livescu, K. Deep canonical correlation analysis. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013; pp. 1247–1255.

18. Tai, K.S.; Socher, R.; Manning, C.D. Improved semantic representations from tree-structured long short-term memory networks. *arXiv* **2015**, arXiv:1503.00075.
19. He, H.; Gimpel, K.; Lin, J. Multi-perspective sentence similarity modeling with convolutional neural networks. In Proceedings of the 20th International Proceedings on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 1576–1586.
20. Wu, Y.; Wu, W.; Xing, C. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *55th Annual Meeting of the Association for Computational Linguistics*; ACL: Stroudsburg, PA, USA, 2017; pp. 496–505.
21. Rajpurkar, P.; Zhang, J.; Lopyrev, K. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *21st International Proceedings on Empirical Methods in Natural Language Processing*; ACL: Stroudsburg, PA, USA, 2016; pp. 2383–2392.
22. Yang, Y.; Yih, S.W.; Meek, C. WikiQA: A Challenge Dataset for Open-Domain Question Answering. In *20th International Proceedings on Empirical Methods in Natural Language Processing*; ACL: Stroudsburg, PA, 2017; pp. 2013–2018.
23. Manning, C. Surdeanu, M. Bauer, J. The Stanford CoreNLP natural language processing toolkit. In *52th Annual Meeting of the Association for Computational Linguistics*; ACL: Stroudsburg, PA, USA, 2014; pp. 55–60.
24. Zhang, R.; Lee, H.; Radev, D. Dependency sensitive convolutional neural networks for modeling sentences and documents. *arXiv* **2016**, arXiv:1611.02361.
25. Zhou, C.; Sun, C.; Liu, Z. A C-LSTM Neural Network for Text Classification. *Comput. Sci.* **2015**, *1*, 39–44.
26. Mikolov, T.; Sutskever, I.; Chen, K. Distributed representations of words and phrases and their compositionality. In *9th International Proceedings on Advances in Neural Information Processing System*; MIT Press: Cambridge, MA, USA, 2014; pp. 3111–3119.
27. Kusner, M.; Sun, Y.; Kolkin, N. From word embeddings to document distances. In *32nd International Proceedings on International Conference on Machine Learning*; ACM: New York, NY, USA, 2015; pp. 957–966.
28. Chung, J.; Gulcehre, C.; Cho, K.H. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* **2014**, arXiv:1412.3555.
29. Pennington, J.; Socher, R.; Manning, C. Glove: Global vectors for word representation. In *19th International Proceedings on Empirical Methods in Natural Language Processing*; ACL: Stroudsburg, PA, USA, 2014; pp. 1532–1543.