Building Task-Oriented Dialogue Systems for Online Shopping

Zhao Yan[§*], Nan Duan[‡], Peng Chen[◇], Ming Zhou[‡], Jianshe Zhou[¶] and Zhoujun Li[§†]
[§]State Key Lab of Software Development Environment, Beihang University, Beijing, China
[‡]Microsoft Research, Beijing, China
[◇]Microsoft Xiaoice Team, Beijing, China
[¶]BAICIT, Capital Normal University, Beijing, China
{yanzhao, lizj}@buaa.edu.cn {nanduan, peche, mingzhou}@microsoft.com {zhoujs}@cnu.edu.cn

Abstract

We present a general solution towards building task-oriented dialogue systems for online shopping, aiming to assist online customers in completing various purchase-related tasks, such as searching products and answering questions, in a natural language conversation manner. As a pioneering work, we show what & how existing natural language processing techniques, data resources, and crowdsourcing can be leveraged to build such task-oriented dialogue systems for E-commerce usage. To demonstrate its effectiveness, we integrate our system into a mobile online shopping application. To the best of our knowledge, this is the first time that an dialogue system in Chinese is practically used in online shopping scenario with millions of real consumers. Interesting and insightful observations are shown in the experimental part, based on the analysis of human-bot conversation log. Several current challenges are also pointed out as our future directions.

Introduction

One goal of Dialogue Systems (DS) is to enable communication between human and computer in natural language, instead of complex commands or procedures. Dialogue systems can be broadly divided into two categories (Su et al. 2016): chat-oriented systems which aim to converse with users and provide interesting and reasonable contextually relevant responses (Banchs and Li 2012; Ji, Lu, and Li 2014; Yan et al. 2016) and task-oriented systems designed to assist users to achieve specific goals (e.g. find products, restaurants or flights) (Henderson, Thomson, and Williams 2014; Bohus and Rudnicky 2009). We propose to build a task-oriented dialogue system for online shopping where the goal is to recommend more relevance products and provide informations about product to user. Our system also support to response chit-chat utterance which makes it more like a human shopping assistant.

A large number of recent task-oriented dialogue systems in both industry (Apple Siri, Microsoft Cortana, Google Now, Amazon Echo, Baidu Duer and Facebook M) and academia (Young et al. 2013; Gupta et al. 2006; Wang, Deng, and Acero 2005; Rieser and Lemon 2010), have focused on developing natural language understanding (NLU) techniques for parsing utterance from user into predefined semantic slots. Developing a structured ontology for NLU can be very difficult in cold-start stage where domain experts have to define the slot and possible values for specific domain. One of the biggest challeng for shopping scenario is that an E-commerce store usually contains hundreds even more product categories. Moreover, there are enormous differences between the semantic slots of various categories. For example, we usually talk about volume for refrigerator but screen size for cell phone. Previous data-driven works for slot induction are mostly based upon human-human dialogue corpus (Tür et al. 2012; Chen et al. 2013; Chen, Wang, and Rudnicky 2014). However, it is hard to collect a large scale human-human dialogue data in shopping scenario.

To deal with the problem we mentioned, our work focus on using three kinds of data resources that are common to most E-commerce web service provider or easily crawled from webs, including: (i) product knowledge base, which is provided by the E-commerce partner and contains structured product information; (ii) search log, which is closely linked with products, natural language queries and user selection behaviors (mouse click); (iii) community sites, where user post their intents in natural language and can be used to mine purchase-related intents and paraphrases of product-related terms. Besides, we show that crowd sourcing is necessary to build such AI bot.

To demonstrate the effectiveness of our proposed method, an AI bot is built as a shopping assistant and embed into a mobile online shopping application of our E-commerce partner. All consumers can communicate with our AI bot via natural language within the application, and the bot tries to either help consumers complete their purchase-related tasks based on session-level understanding of user utterances, or chit-chat with them. By analyzing the human-bot conversation log, we demonstrate interesting findings, point out current issues and future work.

Our proposed approach differs previous work on dialogue system from two aspects:

- **Training data**. Most of previous dialogue system works rely on labeled data as supervision to train statistical models for slot filling, dialog state tracking, police se-

[*]Contribution during internship at Microsoft Research Asia
[†]Corresponding author

lection, and etc. Since such annotated data is scarce for many practical scenarios such as online shopping, we, instead, propose an alternative solution to leverage existing data resources to build task-oriented dialogue systems, together with lightweight crowdsourcing. Note, this doesn't mean our method is cleverer than previous ones, it is just a compromise in the '*cold-start*' stage.

- **Domain scale**. Most of previous dialogue system works focus on restricted domains with pre-defined entities and semantic labels limited in size. In contrast, the size of the domain knowledge base (11M products with 1,080 categories) used in this paper is very large, which brings new challenges to not only algorithms and models, but also product designs. Furthermore, we also propose a pipeline to solve intent mining and detection tasks.

We integrate AI bot with E-commerce in a deep way. To the best of our knowledge, this is the first time that a Chinese AI bot is practically used in online shopping *with millions of real consumers*.

System Formalization

Formally, a task-oriented dialogue system consists of the following four main components:

$$\mathcal{DS} \quad = \quad \{\mathcal{QU}, \mathcal{ST}, \mathcal{DM}, \mathcal{PKB}\}$$

$\mathcal{QU}$ denotes **query understanding**, whose input is $\mathcal{Q}_t$, and output is $\mathcal{M}_t$. $\mathcal{Q}_t$ represents a user utterance issued at time t. $\mathcal{M}_t$ represents the formal meaning representation of $\mathcal{Q}_t$.

$\mathcal{ST}$ denotes **state tracking**, whose input are $\mathcal{M}_t$ and $\mathcal{H}_{t-1}$, and output is $\mathcal{H}_t$. $\mathcal{H}_t$ represents the dialogue state at time t, which is generated by accumulating the meaning representations of utterances within a conversation session.

$\mathcal{DM}$ denotes **dialogue management**, whose input is $\mathcal{H}_t$ and output is a natural language response $\mathcal{R}_t$. The inner function is to select a more suitable action with its natural language expression to respond user based on current dialogue state $\mathcal{H}_t$.

$\mathcal{PKB}$ denotes **product knowledge base**, which is composed of a set of product triples $\langle p, n, v\rangle \in \mathcal{P} \times \mathcal{N} \times \mathcal{V}$ (e.g., $\langle$Huawei P9, DisplaySize, 5.2 inches$\rangle$). $\mathcal{P}$ denotes a set of *products* (e.g., $Huawei\ P9$), $\mathcal{N}$ denotes a set of *attribute names* (e.g., $DisplaySize$), and $\mathcal{V}$ denotes a set of *attribute values* (e.g., $5.2\ inches$).

Query Understanding

Given an utterance $\mathcal{Q}_t$, natural language understanding component generates its representation $\mathcal{M}_t$, which is specially designed for the online shopping scenario:

$$\mathcal{M}_t \quad = \quad \langle \mathcal{I}, \mathcal{C}, \mathcal{A} \rangle$$

$\mathcal{I}$ denotes user's **intent** expressed by utterance $\mathcal{Q}_t$, which determines the action (such as recommendation or QA) that $\mathcal{DM}$ should take;

$\mathcal{C}$ denotes product **categories** that $\mathcal{Q}_t$ is talking about, which determines the possible products that will be considered by $\mathcal{DM}$;

$\mathcal{A}$ denotes product **attribute** which is a set of $\langle$attribute name, attribute value$\rangle$ (or $\langle n, v\rangle$) pairs extracted from $\mathcal{Q}_t$. For example, "*recommend me a Huawei phone with 5.2 inch screen*" will be parsed to a representation $\mathcal{M}$ by $\mathcal{QU}$, where $\mathcal{M}.\mathcal{I} = Recommendataion$, $\mathcal{M}.\mathcal{C} = Cellphone$, $\mathcal{M}.\mathcal{A}.Brand = Huawei$, and $\mathcal{M}.\mathcal{A}.DisplaySize = 5.2\ inches$.

In the rest of this section, we describe how to obtain intention, category and attribute respectively, as query intent detection, product category detection, and product attribute detection tasks.

Query Intent Detection

Utterances mentioning identical product-related terms may have totally different intentions. The system need to decide how to act which dependent on the user's intention. For example, 'Huawei P9' is mentioned by "*I want to buy a P9 phone*", "*How about Huawei P9*" and "*The Huawei P9 is beautiful*". The first utterance need use "Recommand" action and the seconde need to "Qustion Answering" action. The third utterance doesn't contain any purchase intention and need trigger chit-chat engine to response.

Table 1 shows four most-frequent purchase intents we mainly focus on.

Intent Name	Example
Recommendation	recommend me a #
Comparison	# and #, which is better
Ask Opinion	how about #
QA	what is the # of #

Table 1: Intent phrase examples (in English translations), where # denotes a mention of a product-related term.

Intent Phrase Mining User express different intentions usually by explicit phrases. For example "recommendation" by the phrases "*want to buy*" and "ask opinion" by "*how about*". Motivated by this, we define the goal of *intent phrase mining* is to mine high-quality intent phrases, such as "*want to buy*", from raw corpus.

There are several places where people would like to post their purchase-related intents, i.e., search engines, community sites and social network. In this paper, we perform purchase intent phrase mining (Algorithm 1) from questions posted to community sites, as such questions are often natural language sentences with explicit intent expressions. We leave mining intents from other two resources as our future work.

Line 1 uses product-related terms (i.e., product, brand, and category names) to select product-related questions. Line 2 performs *PhraseSegmentation* algorithm to segment sentences in corpus into phrases, which first collects aggregate counts for all phrases in $\mathcal{Q}_D$ that satisfy a certain minimum support threshold, and then employs a bottom-up agglomerative merging to greedily merge the best possible phrases for corpus segmentation. Line 3 runs phrasal-level topic modeling algorithm *PhraseLDA* based on phrase-segmented corpus, which makes sure that all words within a phrase are associated to an identical topic. The detail of

Algorithm 1: Intent Phrase Mining

1 Collect & Filter questions posted to a community site to obtain a set of questions $\mathcal{Q}_D = \{q_1, ..., q_D\}$, each of which should contain at least one product name, brand name or category name contained by a given product knowledge base;
2 Run *PhraseSegmentation* on $\mathcal{Q}_D$ to segment each question q_k into a phrase sequence $\{ph_{k1}, ..., ph_{kn}\}$;
3 Run *PhraseLDA* on phrase-segmented $\mathcal{Q}_D$ to get topic clusters;
4 A set of purchase-related intents is defined by crowdsourcing, based on topic-based phrase clusters;
5 For each intent, a set of phrases is selected by crowdsourcing;
6 Return an purchase intent set $\mathcal{I}$ with labeled intent phrases;

above two phrase-based algorithms are both presented in (El-Kishky et al. 2014). Crowdsourcing is used to define purchase-related intents (Line 4) and select high-quality intent phrases (Line 5) that express specific purchase intents, based on resulting topic-based phrase clusters. These two tasks are done in one day.

We crawl raw questions from Baidu Zhidao[1]. After filtering the full question set based on the product knowledge base, there are 3,146,063 questions left in $\mathcal{Q}_D$. The minimum support threshold is set to 5 for *FrequentPhraseMining*, and the topic size is set to 1,000 for *PhraseLDA*. After labeled by crowdsourcing, there are totally 1,116 distinct intent phrases left, each is labeled by an intent name from the intent set $\mathcal{I}$.

In particular, three '*session-aware*' intents are also considered, each of which depends not only on the current utterance, but also on the current session: (1) **Add Filter Condition**, which denotes the user tries to tell the bot what he/she wants by adding filter conditions in a multi-turn manner, such as "too expensive", "smaller one"; (2) **See-More**, which denotes the user wants to see more products retrieved based on the current filter conditions, such as "another one", "change", "next"; (3) **Negation**, which denotes the user is not satisfied with the current product recommended by do not show why explicitly, such as "not good", "I don't like it";. As such intents are seldom expressed by single-turn questions, we treat them specially, and mine their intent phrases from the end-to-end log data of our AI bot based on the same mining pipeline described in Algorithm 1, and add them to $\mathcal{I}$.

Intent Classification For each intent $\mathcal{I}_k \in \mathcal{I}$, we collect 2,000 questions from $\mathcal{Q}_D$, each of which contains at least one intent phrase of $\mathcal{I}_k$. We also collect 2,000 questions with no purchase intent (chit-chat). All labeled questions are used to train a multi-class classifier to decide which defined intent the utterance contains, or just chit-chat.

Product Category Detection

In this paper, we treat product category detection as a classification task. Given a user utterance with a purchase intent detected, the goal of product category detection is to predict the category of the product that the utterance is talking about. For example, *"I want to buy a dslr"* indicates that the user want to buy a product whose category is "*Camera*". The reason of separating category detection from the detection of other attribute values is that, the total number of products is usually very large (11M products in this paper). Using product category as a filtering constraint, the size of the possible products that are considered by the response generation component can be greatly reduced (thousands of products per each category).

In this paper, we solve this classification task based on a CNN-based approach that resembles (Huang et al. 2013) and (Shen et al. 2014):

- **Input Layer**. Traditionally, each word after tokenization can be represented by a one-hot word vector, whose dimensionality equals to the word size. However, as the size of Chinese words is large, it is expensive to learn model parameters. To alleviate this issue, we represent each Chinese word as a count vector of characters. We then obtain representation of the t^{th} word-n-gram in an utterance $\mathcal{Q}$ by concatenating the character vectors of each word as: $l_t = [w_{t-d}^T, ..., w_t^T, ..., w_{t+d}^T]^T$, where w_t denotes the t^{th} word representation, and $n = 2d + 1$ denotes the contextual window size, which is set to 3.
- **Convolution Layer**. The convolution layer performs sliding window-based feature extraction to project the vector representation l_t of each word-n-gram to a contextual feature vector h_t:

$$h_t = tanh(W_c \cdot l_t)$$

  where W_c is the convolution matrix, $tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$ is the activation function.
- **Pooling Layer**. The pooling layer aggregates local features extracted by the convolution layer to form a sentence-level global feature vector with a fixed size independent of the length of the input utterance. *Max pooling* is used to force the network to retain the most useful local features by $l_p = [v_1, ..., v_K]^K$, where

$$v_i = \max_{t=1,...,T} \{h_t(i)\}$$

- **Semantic Layer**. For the global representation l_p of $\mathcal{Q}$, one more non-linear transformation is applied as:

$$y(\mathcal{Q}) = tanh(W_s \cdot l_p)$$

  where W_s is the semantic projection matrix, $y(\mathcal{Q})$ is the final semantic vector representation of $\mathcal{Q}$.

Given a user utterance $\mathcal{Q}$ and the product category list to be ranked, we first compute semantic vectors for $\mathcal{Q}$ and all product categories using CNN. We then compute a similarity score between $\mathcal{Q}$ and each product category $\mathcal{C}_i$ by measuring the cosine similarity between their semantic vectors:

$$Sim(\mathcal{C}_i, \mathcal{Q}) = Cosine(y(\mathcal{Q}), y(\mathcal{C}_i))$$

The posterior probability of a product category given an utterance is computed based on this similarity score through

[1]http://zhidao.baidu.com, Chinese Community Q&A service

a softmax function, and the model is trained by maximizing the likelihood of the correctly associated product categories given training utterances, using stochastic gradient descent (SGD).

Product Attribute Extraction

Given an utterance $\mathcal{Q}$, the goal of product attribute extraction is to annotate $\mathcal{Q}$ by attribute names and values $\hat{\mathcal{T}} = \hat{t}_1^K$:

$$\begin{aligned}\hat{\mathcal{T}} &= \arg\max_{\mathcal{T}} P(\mathcal{T}|\mathcal{Q}) \\ &= \arg\max_{\mathcal{T}} P(t_1^K|m_1^K) \\ &= \arg\max_{\mathcal{T}} \prod_{k=1}^{K} P(t_k|m_k)\end{aligned}$$

$m_k \in \mathcal{Q}$ is an n-gram, $t_k \in \mathcal{A}$ (or $t_k \in \mathcal{V}$) is an attribute name (or an attribute value) mentioned by m_k, $P(t_k|m_k)$ denotes the probability that m_k can be tagged as t_k. For a word that cannot be tagged as attribute name or value, we tag it as '[*word*]', with a small value assigned to $P([word]|m)$.

How to obtain the triples $Para = \{\langle m, t, P(t|m)\rangle\}$ can be seen as a paraphrase mining task:

- **For attribute values that are product names**, we mine their paraphrases from the product search log data. This is due to the fact that most E-commerce web sites provide site search services to help users to find their desired products within the site. When users are querying via site search engine, they often use product name as queries directly. First, we obtain $\langle q, url_p\rangle$ pairs from the log, where q denotes a query issued to the site search engine and url denotes a product page that is clicked by q and introduces a product whose name is p. Then, we compute $link(p|q)$ as the score that q is an alternative expression of a product name p,

$$\begin{aligned}link(p|q) &= p(p|q) \cdot p(q|p) \\ &= \frac{\#(p,q)}{\sum_{p'} \#(p',q)} \cdot \frac{\#(p,q)}{\sum_{q'} \#(p,q')}\end{aligned}$$

  $\#(p,q)$ denotes the number of times that q clicked a product page whose name is p. Based on $link(p|q)$, we can further compute a distribution:

$$P(p|q) = \frac{link(p|q)}{\sum_{p'} link(p'|q)}$$

  $P(p|q)$ actually denotes the distribution of a query q on a set of product-related terms. Note that, users could type queries other than product names in site search engine, such as brand names, product categories, or a combination of them. Therefore, for each triple $\langle p, q, P(q|p)\rangle$, we only keep it for the usage of product attribute extraction, if $P(q|p)$ exceeds a pre-defined threshold.

- **For all the other attribute values and names**, we mine their paraphrases from various data sources, including aliases of entities stored in knowledge base, the anchor text in Web documents, Wikipedia redirect table, and synonyms mined by a open information extraction method (Hearst patterns).

How to obtain $\hat{\mathcal{T}}$ is solved by a DP-based algorithm, which is shown in Algorithm 2 and Algorithm 3.

Algorithm 2: Product Attribute Extraction

Input: $\mathcal{Q}_t$
Output: $\hat{\mathcal{T}} = \hat{t}_1^K$

1 $\mathcal{T} = \emptyset$;
2 $\mathcal{T}_{nBest} = \emptyset$;
3 $GlobalSearch(0, \mathcal{Q}_t, \mathcal{T}, \mathcal{T}_{nBest})$;
4 Return $\hat{\mathcal{T}} = \arg\max_{\mathcal{T} \in \mathcal{T}_{nBest}} Score(\mathcal{T})$;

$\mathcal{T}_{nBest}$ (Line 2) stores all possible tagging results of $\mathcal{Q}_t$, $Score(\mathcal{T})$ is computed as $\prod_{k=1}^{K} P(t_k|m_k)$, where $\mathcal{T} = t_1^K$.

Algorithm 3: Global Search

Input: $index, \mathcal{Q}_t, \mathcal{T}, \mathcal{T}_{nBest}$
1 **if** $index == |\mathcal{Q}_t|$ **then**
2 $\mathcal{T}_{nBest}.Add(\mathcal{T})$;
3 **end**
4 **else**
5 $m = \emptyset$;
6 **for** $i = index; i < |\mathcal{Q}_t|; i++$ **do**
7 $m = m + " " + w_i$;
8 Add $\langle m, [word], P([word]|m)\rangle$ to $\mathcal{T}$;
9 $GlobalSearch(i+1, \mathcal{Q}_t, \mathcal{T}, \mathcal{T}_{nBest})$;
10 Remove $\langle m, [word], P([word]|m)\rangle$ from $\mathcal{T}$;
11 **foreach** $\langle m_k, t_k, P(t_k|m_k)\rangle \in Para$ **do**
12 **if** $m == m_k$ **then**
13 Add $\langle m_k, t_k, P(t_k|m_k)\rangle$ to $\mathcal{T}$;
14 $GlobalSearch(i+1, \mathcal{Q}_t, \mathcal{T}, \mathcal{T}_{nBest})$;
15 Remove $\langle m_k, t_k, P(t_k|m_k)\rangle$ from $\mathcal{T}$;
16 **end**
17 **end**
18 **end**
19 **end**

Query understanding plays a key role in many AI-related topics. Rule-based methods, such as (Ward 1994) and (Gupta et al. 2006), have disadvantages on scalability and labeling cost. Data-driven methods solve it using various models, such as HMM/CFG (Wang, Deng, and Acero 2005), CRF (Wang and Acero 2006), SVM (Mairesse et al. 2009), RNN (Mesnil et al. 2015), and etc. But most of them rely on labeled data, which is often unavailable to practical domains. We explore an alternate way and use existing resources for the target domain, and leave refinement by labeled in-domain data as next step.

State Tracking

The state tracking component ($\mathcal{ST}$) maintains the dialogue state $\mathcal{H}_t$, which denotes the representation of the dialogue session till time t. The $\mathcal{ST}$ works as following:

1) Updates intention state $\mathcal{M}_t.\mathcal{I}$ based on the function $SessionAwareIntentUpdate(\mathcal{M}_t, \mathcal{H}_{t-1})$ which updates $\mathcal{M}_t$'s intent based on the following rules:

- If $\mathcal{M}_t.\mathcal{I} \in \mathbf{I}$ is not a session-aware intent, then keep $\mathcal{M}_t.\mathcal{I}$ unchanged;

- If $\mathcal{M}_t.\mathcal{I} \in \mathbf{I}$ is a session-aware intent, and $\mathcal{H}_{t-1}.\mathcal{I} =$ *Recommendation*, set $\mathcal{M}_t.\mathcal{I}$ as *Recommendation*;
- Otherwise, set $\mathcal{M}_t.\mathcal{I}$ as *Chit-chat*.

2) If an utterance is detected as chit-chat, then no update is taken, $\mathcal{H}_t = \mathcal{H}_{t-1}$.

3)If the product category of the current utterance is identical to the product category stored in $\mathcal{H}_{t-1}$, then $\mathcal{H}_t$ will inherit all information stored in $\mathcal{H}_{t-1}$. Otherwise, the content of $\mathcal{H}_t$ will be updated totally based on $\mathcal{M}_t$, including category ($\mathcal{M}_t.\mathcal{C}$), intent ($\mathcal{M}_t.\mathcal{I}$) and attributes ($\mathcal{M}_t.\mathcal{A}$).

Note that, if the number of continuous chit-chat utterances exceeds a pre-defined number, or the time interval between two consecutive utterances exceeds a pre-defined length at time t, then $\mathcal{H}_t$ will be cleaned, as a *Forgetting Mechanism*.

Tracking dialogue states is the key to ensure user experience on multi-turn conversation. The main reason that we do not follow previous works (Mrkšic et al. 2015; Henderson, Thomson, and Young 2013) is that, in the current '*cold-start*' stage, we don't have enough data to train statistical models. We leave leveraging session-level labeled data to improve state tracking task as another future work.

Dialogue Management

At each turn in the conversation, the dialogue management component ($\mathcal{DM}$) takes the current dialogue state $\mathcal{H}_t$ as its input, performs different actions based on $\mathcal{H}_t$, and outputs corresponding results as responses. Main actions that are considered in the online shopping scenario include:

- **Recommendation**. This action will be triggered when $\mathcal{H}_t.\mathcal{I}$ is *Recommendation*, and it will retrieve products from product database based on the product category and attribute values detected and stored in $\mathcal{H}_t$;
- **Comparison**. This action will be triggered when (i) $\mathcal{H}_t.\mathcal{I}$ is *Comparison*, and (ii) multiple product/brand names with the same category are detected in $\mathcal{H}_t$. It will compare target products/brands based on their percentages of positive and negative feedbacks given by the consumers and recorded by the E-commerce partner. The response for this intent can be generated based on other resources, and we leave it as another future work;
- **Opinion Summary**. This action will be triggered when $\mathcal{H}_t.\mathcal{I}$ is *Ask Opinion*, and it will summarize the opinions of the target product/brand stored in $\mathcal{H}_t$, based on product review data provided by the E-commerce partner;
- **Question Answering**. This action will be triggered when a product name and one of its attribute names are detected in $\mathcal{H}_t$, but the corresponding attribute value is missing. It returns the missing attribute value by looking up the product database. We treat it as a single-turn KB-QA task, and solve it by the method (Yih et al. 2015);
- **Proactive Questioning**. This action will be triggered when (i) a recommendation intent is detected, (ii) a category is detected, and (iii) no constraint is detected or in $\mathcal{H}_t$. The response is a question, such as "*what kind of* $\mathcal{H}_t.C$ *do you want to buy?*". Such questions are template written by human via crowdsourcing. Note, in the current product design, proactive questioning is only triggered when constraint is missing. As the capability of session-level utterance understanding grows, it can be triggered more for other attributes;
- **Chit-chat**. This action will be triggered when none purchase-related intent can be detected. Motivated by the IR-based response generation methods, such as (Ji, Lu, and Li 2014), this component generates a response to a given user utterance q_t based on the following criterion:

$$\widetilde{r} = \arg \max_{<q,r> \in DB_{QR}} \sum_i \lambda_i \cdot h_i(q, q_t)$$

$DB_{QR} = \{\langle q_1, r_1\rangle, \langle q_2, r_2\rangle, ..., \langle q_M, r_M\rangle\}$ denotes query-response pairs collected from Web, where q_m denotes a query from a user, r_m denotes a response of q_m from another user. h_i denotes the i^{th} feature that measures the similarity between u and q, and λ_i denotes h_i's feature weight. The underlying idea of this component is to: (i) find an existing query $\widetilde{q}$ that is most similar to q_t, and (ii) return $\widetilde{q}$'s response $\widetilde{r}$ as the response of q_t.

	Statistic
# of products	11,082,520
# of categories	1,080
# of attributes	3,359
# of $\langle p, a, v\rangle$ triples	196,853,014

Table 2: Statistics of product knowledge base.

Experiment

Data

The product knowledge base is provided by our E-commerce partner, which covers a variety of the most common products, e.g., household appliances, cellphones, kitchenware, and etc. Table 2 lists some statistics. From Table 2 we can see that, query understanding is very challenging as its size is very large.

The log data from product search engine of E-commerce web site, which contains 6,315,233 $\langle q, url_p\rangle$ pairs, is used for product attribute extraction; The crawled data from Baidu Zhidao, which includes 3,146,063 unique questions after filtering, is used for purchase intent phrase mining.

End-to-End Analysis

We embed our AI bot as a shopping assistant into the mobile shopping application of our partner. Consumers can communicate with our bot via natural language within the application. When a *Recommendation* intent and a product category are detected from consumer utterances, the bot will retrieve related products from product knowledge base, and return them back to the consumers, together with purchase links. Consumers can add additional requirements to refine the products recommended, by multi-turn conversation. The number of active users per day is around 1M.

We sample 28,494 sessions from the conversation log during the 1^{st} week after the bot is released. A **session** denotes all conversation turns between a consumer and robot, before

	Statistic
# of sessions	28,494
# of conversation turns	720,890
# of conversation turns per session	25.3
# of products recommended	84,547
# of products recommended per session	2.97

Table 3: Statistics of sampled human-bot conversation log.

	Statistic
# of chit-chat utterances	1,990
# of non chit-chat utterances	511
# of non chit-chat single-turn utterances	256
# of non chit-chat multi-turn utterances	255

Table 4: Statistics of annotated conversation log.

the consumer leaves the app. A **conversation turn** denotes a consecutive utterance-response pair, utterance from user and response from AI bot. Table 3 shows some statistics. The average number of conversation turns per session is 25.3. We can maintain the conversation for such a long session as we integrate a chit-chat engine, instead of handling purchase-related utterances only.

We annotate 100 sessions (with 2,501 conversation turns), which are randomly selected from 28,494 sessions. Table 4 lists some statistics, where the response of a **single-turn utterance** depends on itself only; the response of a **multi-turn utterance** not only depends on the utterance itself, but also depends on previous utterances. More findings are observed based on this annotated data:

- **Nearly 80% utterances are chit-chat queries.** It shows the importance of the chit-chat component to any AI bot, as we find that most of sessions start from chit-chat utterances. If the robot cannot reply them, then the conversation may not be able to continue. For example, the most frequent queries are like "*Hello*", "*Who you are?*", "*How old are you?*", "*Are your a girl?*", and etc;
- **49.9% utterances depend on previous utterances.** In the online shopping scenario, consumers usually communicate with the bot more than one turns to express their intents and requirements. This brings the challenge to handle multi-turn conversations. For example, "*What size is that?*" is a query with a QA intent, where "*that*" denotes the last product recommended to the consumer;
- **The intent distribution is listed in Table 5.** The most frequent intent is *Recommendation* (50.7%). 55.2% of utterances with this intent contain explicit intent phrases, such as "*recommend me*" or "*I want to buy*"; the others only contain product categories or product names as the whole utterances. The most challenging intent is *Add Filter Condition*. Two reasons to explain why the precision of this intent is low (only 50.5%): (1) product attribute extraction is difficult, as mining paraphrase for more than 11M products is not an easy task. (2) multi-turn conversation is challenging.
- **Chit-chat plays a key role in maintaining the conversation.** When detection or extraction errors happen, users can complain the bot about such mistakes. An interesting observation is that, to a certain degree, chit-chat can reduce consumer's dissatisfaction on the conversation quality, even its content is not relevant to business.

Intent	Proportion	Precision
Recommendation	50.7%	74.1%
Add Filter Condition	21.3%	50.5%
See-More	10.4%	60.4%
Negation	10.6%	66.0%
QA	4.1%	81.0%
Ask Opinion	2.0%	70.0%
Comparison	0.8%	100%

Table 5: Intent distribution and detection accuracy.

Method	ACC@1	ACC@2	ACC@3
BM25	58.7%	76.3%	83.9%
MLR	62.9%	76.8%	81.6%
Naive Bayes	64.8%	77.6%	85.0%
Our Model	67.3%	80.3%	85.4%

Table 6: Evaluation on product category detection (off-line).

Evaluation on Product Category Detection

We evaluate product category detection by two settings. In **Off-line** setting, we first transform each $\langle q, url_p \rangle$ pair (6,315,233 in total) to a $\langle q, \mathcal{C}_p \rangle$ pair by finding p's category from product knowledge base. Next, we split $\langle q, \mathcal{C}_p \rangle$ pairs into a training set (8/10), a dev set (1/10) and a test set (1/10). We use **BM25**, **Naive Bayes** and **Multiple Logistic Regression (MLR)** as our baselines. Evaluation results are shown in Table 6, where accuracy@N(ACC@N) is used as the evaluation metric. In on-line setting, the evaluation is performed on labeled 511 non chit-chat utterances only, with results listed in Table 7. Precision, recall and F1 are used as the evaluation metric.

By comparing Table 6 and Table 7, we find: (i) Our model performs significantly better in the On-line setting than in the off-line setting. (ii) In off-line data, there is a huge gap between ACC@1 and ACC@2, but small differences between ACC@2 and ACC@3. The main reason is that very similar categories pairs are exist in the product knowledge base, such as *cellphone accessories* and *cellphone cases*. In the on-line setting, human incline to judge all related categories are related to the query. However, in the off-line setting, each query has only one right category which is automatically labeled by click information.

Evaluation on Utterance Type Classification

Classifying which type an utterance belongs to (either chit-chat or non chit-chat) is important to task-oriented dialogue

Method	Precision	Recall	F1
Our Model	89.8%	67.7%	77.2%

Table 7: Evaluation on product category detection (on-line).

TRUTH	PREDICTION		Recall
	Chit-chat	Non Chit-chat	
Chit-chat	1,878	112	94.4%
Non Chit-chat	133	378	74.0%
Precision	93.4%	77.1%	

Table 8: Evaluation on utterance type classification.

system. We evaluate the quality of utterance type classification in Table 8 based on the method described in purchase intent classification part and list results in Table 8. We find the classification precision on non chit-chat utterances (77.1%) is much lower than the number on chit-chat utterances (93.4%). Most errors are caused by the utterances which contain some intent phrases but are actually chit-chat ones. One of the reason is sampling negative cases is difficult. We leave how to generate more similar negative data for this task as our future work, as more session data will be obtained from log data and labeled as training data.

Conclusion

In this paper, we present a general solution towards building task-oriented dialogue systems for online shopping, aiming to assist online consumers via natural language conversation. To the best of our knowledge, this is the first time that an AI bot in Chinese is used in online shopping scenario, with millions of real consumers. We present end-to-end analysis based on conversation log, and show some findings and statistics. We also point out several current challenges as our future directions.

Acknowledgments

This work was supported by Beijing Advanced Innovation Center for Imaging Technology (No.BAICIT-2016001), the National Natural Science Foundation of China (Grand Nos. 61370126, 61672081), National High Technology Research and Development Program of China (No.2015AA016004), the Fund of the State Key Laboratory of Software Development Environment (No.SKLSDE-2015ZX-16).

References

Banchs, R. E., and Li, H. 2012. Iris: a chat-oriented dialogue system based on the vector space model. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL) System Demonstrations*, 37–42.

Bohus, D., and Rudnicky, A. I. 2009. The ravenclaw dialog management framework: Architecture and systems. *Computer Speech & Language* 23(3):332–361.

Chen, Y.-N.; Wang, W. Y.; Rudnicky, A.; et al. 2013. Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, 120–125.

Chen, Y. N.; Wang, W. Y.; and Rudnicky, A. I. 2014. Leveraging frame semantics and distributional semantics for unsupervised semantic slot induction in spoken dialogue systems. In *IEEE Spoken Language Technology Workshop*.

El-Kishky, A.; Song, Y.; Wang, C.; Voss, C. R.; and Han, J. 2014. Scalable topical phrase mining from text corpora. *Proceedings of the International Conference on Very Large Data Bases (VLDB) Endowment* 8(3):305–316.

Gupta, N.; Tur, G.; Hakkani-Tur, D.; Bangalore, S.; Riccardi, G.; and Gilbert, M. 2006. The at&t spoken language understanding system. *Audio, Speech, and Language Processing, IEEE Transactions on* 14(1):213–222.

Henderson, M.; Thomson, B.; and Williams, J. 2014. The second dialog state tracking challenge. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 263.

Henderson, M.; Thomson, B.; and Young, S. 2013. Deep neural network approach for the dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, 467–471.

Huang, P.-S.; He, X.; Gao, J.; Deng, L.; Acero, A.; and Heck, L. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*, 2333–2338.

Ji, Z.; Lu, Z.; and Li, H. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*.

Mairesse, F.; Gašić, M.; Jurčíček, F.; Keizer, S.; Thomson, B.; Yu, K.; and Young, S. 2009. Spoken language understanding from unaligned data using discriminative classification models. In *Acoustics, Speech and Signal Processing (ICASSP), 2009 IEEE International Conference on*, 4749–4752.

Mesnil, G.; Dauphin, Y.; Yao, K.; Bengio, Y.; Deng, L.; Hakkani-Tur, D.; He, X.; Heck, L.; Tur, G.; Yu, D.; et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on* 23(3):530–539.

Mrkšic, N.; OSéaghdha, D.; Thomson, B.; Gašic, M.; Su, P.-H.; Vandyke, D.; Wen, T.-H.; and Young, S. 2015. Multi-domain dialog state tracking using recurrent neural networks. *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL) Volume 2: Short Papers* 794.

Rieser, V., and Lemon, O. 2010. *Natural Language Generation as Planning under Uncertainty for Spoken Dialogue Systems*. Springer Berlin Heidelberg.

Shen, Y.; He, X.; Gao, J.; Deng, L.; and Mesnil, G. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web (WWW)*, 373–374.

Su, P. H.; Gasic, M.; Mrki, N.; Barahona, L. M. R.; Ultes, S.; Vandyke, D.; Wen, T. H.; and Young, S. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. In *Meeting of the Association for Computational Linguistics (ACL)*.

Tür, G.; Jeong, M.; Wang, Y.-Y.; Hakkani-Tür, D.; and Heck, L. P. 2012. Exploiting the semantic web for unsupervised

natural language semantic parsing. In *Proceedings of the INTERSPEECH conference*.

Wang, Y.-Y., and Acero, A. 2006. Discriminative models for spoken language understanding. In *INTERSPEECH*.

Wang, Y.-Y.; Deng, L.; and Acero, A. 2005. Spoken language understanding. *Signal Processing Magazine, IEEE* 22(5):16–31.

Ward, W. 1994. Extracting information in spontaneous speech. In *Third International Conference on Spoken Language Processing*.

Yan, Z.; Duan, N.; Bao, J.; Chen, P.; Zhou, M.; Li, Z.; and Zhou, J. 2016. Docchat: An information retrieval approach for chatbot engines using unstructured documents. In *Meeting of the Association for Computational Linguistics*.

Yih, W.-t.; Chang, M.-W.; He, X.; and Gao, J. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*.

Young, S.; Gasic, M.; Thomson, B.; and Williams, J. D. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE* 101(5):1160–1179.