Emory IrisBot: An Open-Domain Conversational Bot for Personalized Information Access

Ali Ahmadvand,[*] Ingyu (Jason) Choi, Harshita Sahijwani, Justus Schmidt, Mingyang Sun, Sergey Volokhin, Zihao Wang,[†] Eugene Agichtein[‡]

Emory University
Department of Computer Science
Atlanta, GA, USA

Abstract

We describe IrisBot, a conversational agent that aims to help a customer be informed about the world around them, while being entertained and engaged. Our bot attempts to incorporate real-time search, informed advice, and latest news recommendation into a coherent conversation. IrisBot can already track information on the latest topics and opinions from News, Sports, and Entertainment and some specialized domains. The key technical innovations of IrisBot are novel algorithms for contextualized classification of the topic and intent of the user's utterances, modular ranking of potential responses, and personalized topic suggestions. Our preliminary experimental results based on overall customer experience ratings and A/B testing analysis, focus on understanding the contribution of both algorithmic and surface presentation features. We also suggest promising directions for continued research, primarily focusing on increasing the coverage of topics for in-depth domain understanding, further personalizing the conversation experience, and making the conversation interesting and novel for returning customers.

1 Background and Overview

Our goal is to develop a conversational agent that helps the user be informed about the world around them, while being entertained and engaged. Our envisioned social bot, IrisBot, will incorporate real-time search, informed advice, and latest news recommendation into a fluent and coherent conversation. IrisBot aims to discuss and share information on relevant latest topics and opinions in the News, Sports, Entertainment, and general knowledge, by accurately detecting user's intent, both explicitly stated and implied from the conversation context. Finally, the information will be made more engaging and entertaining by incorporating humor and detecting emotional cues from the user's utterances. To accomplish this, IrisBot aggregates search and recommendation over a variety of information sources, and employs proactive recommendation strategies, simulated emotion, and incorporate sequence to sequence learning to optimize conversation fluency. At the end of the conversation, our goal is for the users to be informed by in-depth discussions of topics of interest, while entertained through expressions of empathy, humor strategically placed to liven up the conversation.

1.1 Background

Conversational agents have been an active area of research for decades, starting from domain-specific, rule-based systems such as ELIZA [24] and PARRY [7]. More general task-based dialogue modeling

[*]Student authors are in alphabetical order.

[†]Student team leader. Email: `zihao.wang2@emory.edu`

[‡]Faculty advisor. Email: `eugene.agichtein@emory.edu`

approaches ([17], [2], [10], [3], [18]) have been explored to break down larger tasks into smaller tasks, or to extract and compile prior knowledge from dialogue corpora. These models nevertheless require extensive effort to customize and configure for each domain and knowledge source, and were not designed for conversations in open domains.

With the availability of large amounts of dialogue and social data online, data driven systems have shown promise. Reinforcement learning based systems ([13], [26], [12], [27], [25]) and deep learning based systems ([20], [4], [14]) have been both extensively studied and applied. These systems have the advantage of adapting to new domains by collecting corresponding datasets and retraining the model. However, these systems still closely rely on training corpora or restricted information, without the ability to query external data sources, thus limiting their ability for informative conversation or conversational search. Unfortunately, fully automated domain-independent conversational agents are still beyond the published state of the art.

1.2 Overview

For this 2018 Alexa Prize challenge our main goal has been to support *conversational search*. Recently, a theoretical framework was proposed to describe the variety of interactions and feedback available in conversational search [16], which provided a valuable framework for subsequent studies. Unfortunately, due to the inherent technical challenges in fielding an open-domain conversational search system, previous empirical studies have been limited (e.g., [21], Avula et al.[1] and Trippas et al.[19]). Thus, fielding an open-domain conversational search system such as IrisBot for live customer traffic goes far beyond the existing state of the art in conversational search. This year we redesigned from ground-up our entry in last year's AlexaPrize 2017 competition[23]. Our new IrisBot system aims to perform well in a realistic, open-domain, effectively unrestricted conversational setting, while still engaging and entertaining customers to support non-information-seeking aspects of conversations.

Our main technical innovations in this Challenge are the novel solutions to the following critical research problems:

- Topic and intent classification of utterances, using conversation context (Section 3).
- Response ranking by aggregating context and classifier signals with response scores (Section 4).
- Contextual, personalized, and cross-component recommendation (Section 6).

We first describe the Irisbot architecture and implementation next (Section 2). Then we describe in detail the main technical innovations: the topic and intent classifier (Section 3), the dialogue manager and response ranking models (Section 4), and topic suggestion algorithms (Section 6). We then overview the main system domain components that perform domain-specific intent classification and response ranking of their own (Section 5). The classification and response ranking algorithms were evaluated through extensive offline experiments in with live customer traffic during the semi-finals period (Section 7), which demonstrate the promise of our approach. We discuss the results, lessons learned, and outline current and planned follow-up research in Section 8, which concludes the paper.

2 IrisBot System Architecture and Description

IrisBot is designed through loose coupling of domain-specific components that interact through the Dialogue Manager. Each utterance is processed to identify the key entities, topics, and intents (as described below), and is heavily annotated with NLP information helpful for the domain specific components to respond to the query. We chose to do "lazy" response evaluation in that the final response ranking is performed after each component returns a candidate response, at which point the responses are ranked and selected based on the combination of the utterance interpretation and the estimated response coherence and relevance. Thus, each query is processed by every component, which has a computational drawback of significant processing for each utterance, and the benefit of redundancy and multiple available fallback responses. We chose to optimize the expected response quality from having redundant options available. Each domain component implements a common set of interfaces, and is expected to return a score of the response, the response type and topic, and follow-up suggestion (which could be the same component or a switch to another topic). As a result, adding new components turned out to be quite easy with the main challenge being to expand the topic classifier to identify when the new component is relevant.

2.1 Architecture

To implement our proposed framework and remain responsive to the customer queries, we constructed a distributed system, constituted by Amazon Lambda function, a load balancer that we created, and actual IrisBot application instances on Amazon EC2. To support up-to-date and fresh content, we also used a centralized data storage server which was updated often and replicated the fresh content to all IrisBot application instances. The overall IrisBot architecture is outlined in Figure 1.

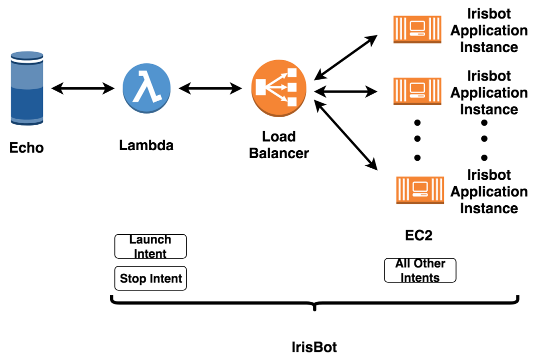

Echo
Lambda
Load Balancer
Irisbot Application Instance
Irisbot Application Instance
Irisbot Application Instance
EC2
Launch Intent
Stop Intent
All Other Intents
IrisBot

Figure 1: IrisBot Distributed Architecture.

The IrisBot Engine (running on each of the IrisBot application EC2 instances) communicates with customers through Amazon Lambda function which in turn communicates with an Echo device. Each engine has the multi-threading capability to handle multiple conversations at the same time, with a global context variable that maintains the context of each conversation. The customers' utterances are transmitted to the IrisBot Engine by the load balancer, which tracks conversations, manages health and availability monitoring and as necessary stops or re-starts traffic on each instance automatically.[4] After processing, the best response from IrisBot Engine is returned back to the customer. Through the analysis of the conversations from August 1st to August 15th, 2018, the average response latency is 0.66 second, the 50th percentile latency is 0 second and the 90th percentile latency is 1 second. The latency is always less than 5 seconds since the time-out threshold is set to be 5 seconds for the system, within which a relevant or a general default response will be returned. The latency has improved over the period from July 16th to July 31st, which has the average latency of 0.74 second, the 50th percentile 0 second and 90th percentile 2 seconds (Latency time is stored as discrete integers, from 0 to 5 seconds).

2.2 Dialogue Manager and Response Selection Overview

The IrisBot Dialogue Manager processes utterances to generate responses using four main execution steps, which are Pre-processing, Task execution, Post-processing and background information pre-fetching as outined in Figure 2(a).

Pre-processing includes NLP processing and component task assignment. The initial NLP processing is a multi-threaded process, running hierarchical topic/intent classification, entity recognition, co-reference resolution, sentiment analysis, and other language understanding tasks. The results of the language understanding process features are stored into context, enabling dialogue manager and other components to easily access and utilize these features.

After the pre-processing step, the dialogue manager aggregates this evidence and sends the utterances in parallel to each domain component. Each component works independently to generate a response, and notifies the dialogue manager when finished.

[4]The automatic restart functionality was implemented after a disaster day late in the semi-finals period, in which two of the application instances were non-responsive at the same time, and were repeatedly returning error messages to the customers. Needless to say, customers were not happy about this. Our new load balancer now monitors both instance availability and basic "health" by checking automatically for common error responses.

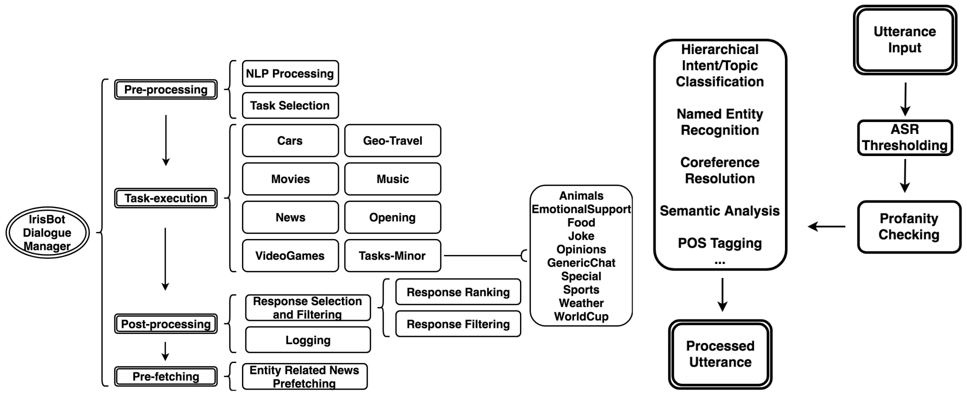


Figure 2: IrisBot Dialogue Manager (a). NLP Processing (b).

In post-processing, the response ranker filters and ranks the responses by NLP and contextual features, including classification probabilities, response scores and stickiness score (an indicator that suggest the extent to which the conversation would remain in the same topic domain). By integrating all these features, it may correct errors produced by the classifier or the process of response generation by components. After IrisBot returns the best response to the customer, pre-fetching process is executed while we wait for a customer to think and reply to IrisBot. Pre-fetching is helpful because we can pre-compute relevant information, retrieve external information (e.g., Web Search results or News articles) and speed up the overall execution time for the next turn. For the current system, pre-fetching is used to extract fact recommendations that are related to the response just returned to the user. Lastly, we have a strict timeout-control mechanism to always return a response within the available time limit (currently set to maximum of 5 seconds, even though the average response time is about 1 second).

Injecting liveliness and humor into responses : Our focus groups and free-form customer feedback alerted us to the fact that our bot sounded too "monotone". This meant it didn't matter that our content was good if we sounded too boring. We addressed this concern by modifying the default speech generation of the responses by using the Amazon Alexa's SSML (Speech Synthesis Markup Language) capabilities. Specifically, we created an internal sounds library which was usable by all components that included a common set of interjections such as "Awesome!". This allowed our bot to sound somewhat consistent while livening up the conversation and not sound monotone and boring. To diversify the responses, we created 'classes' of interjections such as 'positive exclamations', 'sad', 'acknowledgement', etc. We used these classes to diversify the appropriate interjections so that the customer experience had some variety. We furthermore developed our own pitch and rate modifications using the SSML prosody tag to convey excitement, hesitation and emphasis, allowing the bot to convey appropriate emotions to the customer.

2.3 Language Understanding and Entity Recognition pipeline

IrisBot's NLP pipeline is executed during the pre-processing stage in the Dialogue Manager. Many different functions, including both heavy and light NLP modules are executed in parallel to reduce latency. Once each thread is finished, its output is stored in the utterance context maintained by the Dialogue Manager, enabling easy backtracking for components to make smarter decisions.

Knowledge-Based Named Entity Recognition : To recognize key entities we created an index of 3.5 million entities and their types. That includes all the entities that are present in our relational ontology, and all the entities that our domain-specific components can currently handle. Given an utterance, we use it to query our index and and retrieve all the entities present in it and their types.

Relational Ontology : IrisBot has its own knowledge-base that is modeled as a graph, and stores attributes of and relations between millions of entities in the index. It has been initially derived from DBPedia [11] and extended with the entities important to our components. Given an entity, the ontology server allows us to look up entities related to it, for example for a movie it might be a directory and cast list, or for a person it might be a spouse or a place of birth, and many other relations as defined by DBPedia.

Co-reference resolution Co-reference resolution in a dialogue setting becomes challenging because our system expects continuous influx of many different types of entities. It also remains uncertain whether we should prioritize entities appearing on our responses or on user's utterance. For IrisBot, co-reference resolution depends on knowledge-based named entity recognition, along with two main parameters: half-life and bias. Based on half-life, co-reference module continuously discounts the confidence score as time passes, prioritizing newly appeared entity to selection process. Bias is used to control weights between entities appeared on utterances and responses. We tuned these parameters over our development period, and they are incorporated with handcrafted tie-breaking features to assist selection process. Our future work is to incorporate contextual features to candidate selection and enable longer co-reference chains to reach earlier utterances.

2.4 Domain-specific Components

IrisBot currently has a total of 12 domain-specific components, listed below. This is a fluid list as components are added and occasionally retired (e.g., the WorldCup component was retired after the conclusion of the 2018 FIFA World Cup). These components are described in more detail in Section 5 below.

Movies and Shows	Music and Concerts
News	Travel and Activities
Animals	Video Games
Cars	Sports
Opinions	Wikipedia
World Cup	Emotional Support

Table 1: Domain-specific retrieval components

3 Topic and Intent Classification

We chose to follow the current state of the art and the Amazon practice, and to identify both *topic* and *intent*. For example, a customer may request an opinion or recommendation (intent) for a particular movie (topic). We developed our own state of the art classifiers for this task, as described next, as our component capabilities and boundaries did not easily map to available classes provided by existing Amazon Comprehend services and similar. However, we do use these services as input to our own classification.

3.1 Contextual Topic Classification

Our proposed model for contextual topic classification consists of three phases: Independent Topic Classification, Topic Merging, and Contextual Topic Merging.

Phase 1: Independent Topic Classification: For this step we use only the current utterance. The result of this step is a nonexclusive classification vector. For topic classification, a Mixture of Experts Model is applied, which consists of three different classifier that have been trained on different datasets and also with different parameters and architectures. The topic classifier includes a customized classifier, an entity classifier, and an amazon annotator. The latency of the Mixture of Experts Model depends on the classifier that takes the longest time to classify an utterance. In our system, the slowest classifier takes less than 1 ms; therefore, a sequential model takes at most 3 ms to complete the classification task.

Phase 2: Topic Merging from Mixture of Experts: In this step the outputs of three classifiers are merged to produce a single topic distribution based on evidence from all of the "expert" base classifiers.

Phase 3: Contextual Topic Resolution: The third step is Contextual Topic Merging, where the topic distribution for the previous utterances is used as input to inform the final topic classification result. To apply contextual merging, a transition matrix between topics is computed based on the conversation logs, which indicates the transition probability from previous topic to the current one.

The logic and motivation for using the Mixture of Experts Model for this task comes from the "No Free Lunch Theorem", from machine learning, which suggests multiple classifiers for decision making instead of a single powerful model[15]. Moreover, according to the promising results for Convolutional Neural Networks (CNN) models in text classification, we used CNN as the base classifiers [29][9]. Finally, we used a Mixture of CNN models and the Amazon Annotator (as another expert) for the Topic Classification step. The three different models are, 1. our own customized classifier based on the CNN in diagram below, an entity-based classifier, and Amazon Annotator, all or which have been used in this Mixture. The customized classifier has been trained on our customized dataset and the word embeddings has also been tuned on this data. The entity classifier has been trained on DBPedia dataset to find a relation between each entity and the corresponding topic. Consequently, in the Topic Merging step we merge all the results from these three classifiers. We used topic distributions to disperse the probabilities between different topics.

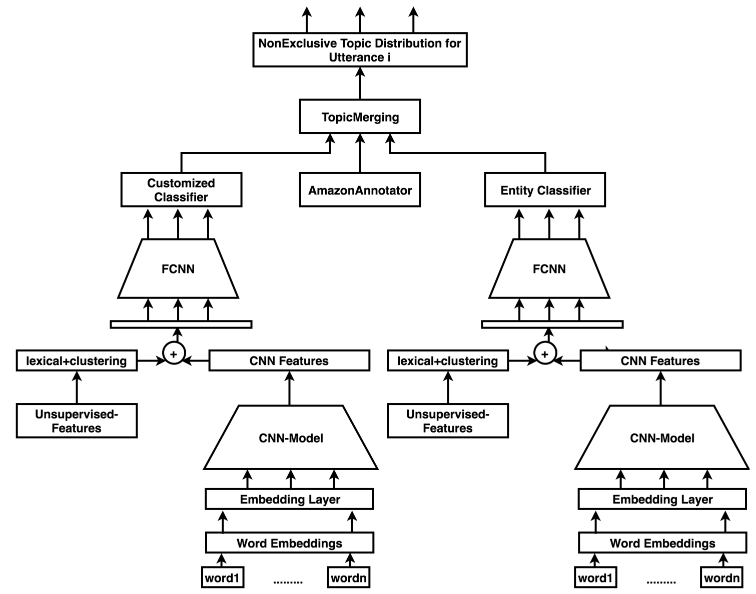


Figure 3: Mixture of Experts Model.

3.1.1 Semantic and Lexical Classification using Convolutional Neural Networks (CNN)

CNN models are used as basic classifiers for both Customized and Entity classifiers, because of promising results in text classifications in the recent years. The basic classifier is a CNN model consisting of 4 layers, an embedding layer of size 300 followed by 3 Convolution layers, any of which consists of a convolution and a max pooling layer. A Fully Connected Neural Networks (FCNN) was used for the final classification. In each Convolution layer, 256 filters of size 2, 3 and 4 were used, respectively. We used l2-regularization of 0.01 and a dropout of 0.5 to avoid over fitting during training. Lexical and unsupervised features were extracted from text utterances and concatenated with CNN features to utilize the CNN features. The final vector was used as input of the FCNN model for classification.

Customized Classifier: we generated more that 50,000 utterances which are customized to our bot, and the components that the bot supported. Google Word2Vec is used for initialization of each word embedding. Then, each word has been trained during training to be more customized to our bot. To use maximum possible number of vocabularies for conversation data, 4 different popular dialogue or conversation datasets such as Cornell, ubuntu and scenarios, and reddit, in addition to our dataset. Finally, more than 25,000 frequent words were extracted from these datasets and were been used as lexical features. To leverage training, we also extracted different unsupervised features like LDA and cosine similarity between LDA topics, to extract high level clustering features from the utterances.

Entity classifier: The architecture for this classifier is the same as the previous classifier, but this classifier uses one more Convolution layer; and 256 filters in each Convolution layer. Finally, a FCNN was applied for final classification. External features such as cosine-similarity and LDA and LSI features had been used as unsupervised features [22][6]. DBpedia entities were used to generate the dataset. Some general rules were applied to assign every entity to a special topic. E.g. University Names can be categorized in both News and Tech topics. Each entity is considered as one word and average Google Word2Vector is used as word embeddings. This dataset contains millions of samples, as a result the generated dataset is much larger than the customized dataset. We used 3M Google word2vector as vocabularies to train this model. The word vectors are not retrained during training to make model more generalized.

Contextual topic post-processing: Contextual information plays an important role in conversational AI. To leverage our model using contextual information in a real conversation, contextual Topic Merging phase is proposed. This phase contains two sub steps. First, specific contextual topic merging and second global contextual topic merging. In the specific contextual topic merging, different heuristics based on our domain knowledge about a specific topic are used to refine the probabilities for a special topic. For example, specific regular expressions were used to capture some corner cases, which the proposed Topic Classifier could not classify them correctly. In the global step, a transition matrix was used which had been computed based on the conversation logs. This matrix represents the transition probabilities between different topics.

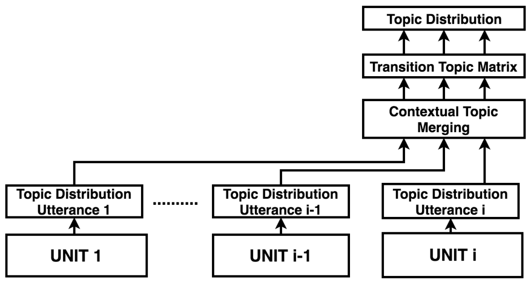


Figure 4: Contextual Topic Merging.

3.2 Intent Classification

Building on the Amazon intent ontology, we considered 11 different intent classes, such as Opinion-request, opinion-delivery, info-request, user-instruction, personal-delivery, topic-switch, topic-acceptance, DontKnow, repetition, self-harm, clarification and user-correction (to support ASR) as our intent classes. The classes of TopicAcceptance and UserCorrection have been added to the original classes from Amazon. Also, the information-delivery intent is the main (default) intent supported by domain-specific components, thus did not need to be predicted with the intent classifier.

Similar to the Topic Classifier architecture, for the intent classifier a 4-layer CNN model was used. For word embeddings, Google Word2Vec for representing the words, and 512 filters in each step are used. Whole 3M vocabularies are used and word vectors did not retrain during training. We also used transfer learning to improve the generalization of the model. For this purpose, TensorflowHub features are used as an external feature [5]. TensorflowHub represents a sentence in the vector space. Those vectors are extracted based on the pre-trained models that had been trained on different domains and large corpus of data. Moreover, another embedding layer of size 16 was added to train POS features, and they have been trained during training. Finally, 37 different POS tags from NLTK have been trained.

The dataset for intent classifier is highly unbalanced, which makes training challenging and might cause over-fitting on the more popular classes such as InformationRequest. To solve this problem, we created balanced training dataset, and used the output of the topic classifier as an input to provide additional contextual features for intent classifier. Thus, pipeline of Topic and Intent classifiers is ultimately used for intent classification. For some classes, such as user-correction and repetition (user asking for repetition), which related to previous utterances, we designed a Post Intent Merging phase, similar to the Contextual Topic Merging described above. In this phase, contextual features are taken into account to decide and potentially modify the intent decision to a more likely one given the conversation context. This is done primarily to deal with important special cases such as recovering from an error, or remaining in a component for following up to a question asked to a customer.

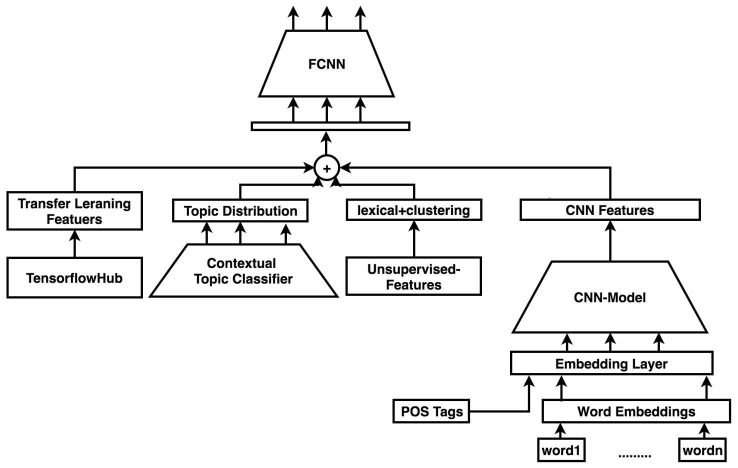


Figure 5: Intent Classifier.

4 Dialogue Manager and Response Ranking

The Dialogue Manager is at the heart of the IrisBot engine. It manages the conversation context, which stores all the NLP annotations, classification results, and is passed to each domain component to record their responses. The context object is used for ranking and selecting the responses and post-processing them before returning to the customers.

4.1 Dialogue Manager: Implementation

As shown in Figure 2, the dialogue manager is a pipeline of utterance processing for natural language understanding (NLU), component task execution, response ranking and filtering, and relevant information pre-fetching.

The utterance processing includes utterance ASR thresholding, profanity checking, and multi-threaded NLP processing. The ASR thresholding checks on the ASR probability input on each possible speech transcription and compares the largest probability with an empirical threshold. The conversation goes to an utterance ambiguity state and the user would be requested to repeat the utterance if the probability is less than the threshold. Profanity check is applied on the utterance and a default answer and a deflection suggestion would be returned if the utterance is detected profane. If the utterance passes the previous two checkpoints, a multi-processing NLP processing is executed to extract features in context for later decision makings. For each NLP function, a thread is designated with a processing timeout limit of 1.5 seconds. The total processing time limit of all threads is 2 seconds.

The features extracted by NLP processes are maintained in the conversation context. Based on context information, the dialogue manager then selects components to take in context and retrieve responses. The are 6 scenarios, as shown in Table 2, for the dialogue manager to make different decisions. For example, if the user intent is Repetition, only the Special State Component will be selected to repeat the last response. If the user intent is Information Request, all domain components would have an equal chance to retrieve responses. Domain components retrieve responses in parallel threads sharing context information with a timeout limit of 2 seconds for each thread and 2.5 seconds in total.

Scenario	Decision
Instruction Request	Special State Component
Pause and Hesitation	Special State Component
SelfHarm	Special State Component
Stop	Special State Component
Opening and Greeting	Conversation Opening Component
Repetition	Special State Component
Information Request	Domain Components

Table 2: Conversation States and Component Assignment for Retrieving Responses.

Assigned by the dialogue manager, components retrieve responses, assign them with relevance scores and follow up scores, and pass them to the response ranking function. The ranking function assigns each response with a final ranking score. The final step is profanity check on the candidate responses and the best response satisfying all requirements will be returned.

Pre-fetching is a separate thread during the period between returning the response back to a user and waiting for the next input from the user. It extracts information from the responses, such as entities, and pre-fetches relevant information for the next turn of conversation.

4.2 Response Ranking: Model and Algorithm

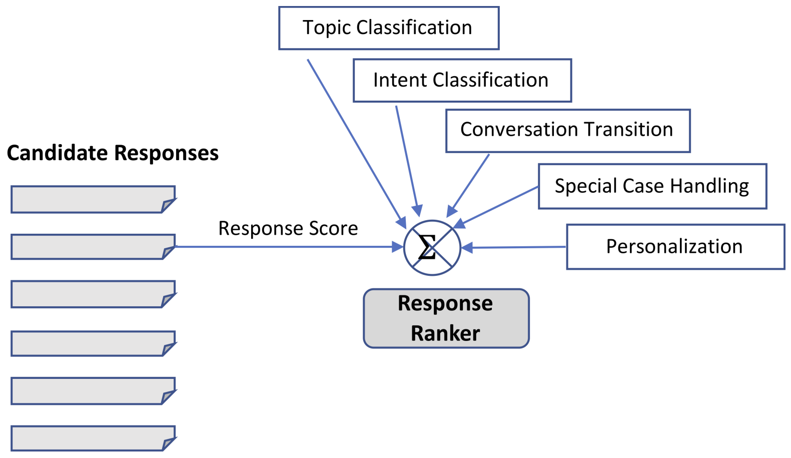


Figure 6: IrisBot Contextual Response Ranker with a hybrid weighted and heuristic ranking strategy.

IrisBot postpones the decision for selecting a response until the candidate responses are returned by the DialogueManager tasks. Instead of only relying on the topic and intent classifier, the response ranker takes into account a variety of response and context features, and handles additional domain-specific cases and states, as illustrated in Figure 6. Specifically, for special cases, especially continuing on the same topic as before, or recovering from an error condition or responding to profanity, the ranker uses pre-defined strategy. For all other cases, a weighted combination of topic and intent classification scores are combined with the response scores, and other contextual information encoded as features for the ranker. This combination of features for ranking the responses is more robust than just relying on one feature such as the result of topic/intent classification. The weighted scoring linear combination model was tuned based on domain knowledge and development data, but can be easily extended to use additional features and more sophisticated learning-to-rank models, trained over the data from the conversations in the Semi-finals.

The most important special cases handled by the ranking model include:

- Component Follow-up: in order to maintain conversation flow, the response ranker prioritizes responses from the most recently active domain component, which may have asked the person a question on previous conversation turn, and is now expecting to provide a follow-up response. For example, if person is speaking to the bot about Movies, and asked the customer about their favorite genre, the response from Movies about the customer's genre utterance will be prioritized. However, if the customer response is on another topic, the normal weighted ranking flow will be used.

- Special state handling: in case of ASR confusion or other error such as component response timeout, the ranking model attempts to select the next most appropriate response, followed by a topic suggestion.
- Dealing with rejection: topic suggestion: a common case is that a person rejects a topic of conversation proposed by the system. In that case, the ranking strategy will recommend another topic. This recommendation can be random, deterministic (following a pre-defined conversation "script", or personalized: prioritizing topics predicted to be interesting to the customer. In Section 7 we empirically compare these strategies for topic suggestion.

5 Domain-Specific Components

We now describe the domain-specific components which allowed the customer to engage the bot in more depth on a subset of topics. Specifically, we decided to build conversations around the topics of Movies, Music, News, Travel, and general information search. Based on the analysis of customer interests we also added components for the topics of Cars, Video Games, Animals, Food, and Sports. Unfortunately, at the time of this report, we have not finished implementing components with knowledge about Literature and Fashion, but plan to include those components in IrisBot by late August 2018, along with other planned enhancements described in Section 8.

5.1 Movies and Shows Component

The Movies component is able to hold in-depth conversations on most entertainment-related topics, offer personalized recommendations, and can direct the customer to other related topics. To accomplish this, this component uses a variety of question and answer templates (which we call "chat-mode") to provide the framework of a natural conversation, as well as determine customer preferences to deal with the initial "cold-start" problem. However, we found that to keep the customer engaged, we must also share immediately useful or entertaining information instead of just asking questions. The general pattern is asking a question, acknowledging the customer's response, providing a recommendation, and asking a related (more in-depth) question to provide an even more interesting or relevant recommendation on the next turn. When the templates are exhausted, the component switches to pure recommendation mode, by offering what the model estimates to be relevant or interesting recommendations - and continue in this mode until the customer indicates that they wish to switch the topic.

Implementation notes: The primary data for movie recommendations was continuously crawled from RottenTomatoes, and included the movie genres, actors, plot, ratings, etc. The data was then indexed using the Python *whoosh* library, to create 2 separate indexes: "fresh movies" (specifically the ones currently in theaters) and the rest. Crawling was performed continuously in the background, updating the indexes nightly. In addition, contents of IMDB were downloaded to provide the list of important entities (people, TV series) to also be able to discuss shows and series.

For each utterance, domain specific (more fine-grained) classifier was implemented using decision trees, which further classified a movies-related utterance to determine if the customer mentioned a movie genre, an actor, or a movie title, along with several other topics. Once the utterance was classified, a query was composed and response generated to return the highest-scoring answer. The score was computed by combining the topic and intent classification described above, combined with match similarity and other domain-specific heuristic criteria. The actual response was always followed by a question, randomly chosen from a recommendation "script" template to offer another recommendation or to ask for the customer opinion or preference.

Personalized recommendation: Based on the customer responses to the "chat-mode", we the movie component could provide increasingly personalized and accurate recommendations of movies. As IrisBot often starts with offering to discuss during the first few turns of the conversation, a large majority of *returning* customers have had some history of interaction with Movies. We used that history to provide already personalized suggestions, based on entities they previously liked, such as a favorite genre, actor, or director.

5.2 Music Component

Music was the second most popular topic of conversations with IrisBot. In its interaction model and implementation, Music is similar to the Movies component described above. This component provides answers to many music-related questions, offers personalized recommendations, and can hold general music conversations with the customer with the goal of making a more relevant music recommendation or providing interesting information. Just as Movies component does, the Music component also attempts to chat with the customer to gather indicators of interest for music genres or sub-genres, artists, and attempts to offer similar artists, genres, or provide general information of interest. Initially the Music component was focused on concerts, until it became clear that the majority of customers were not interested, and the topics and recommendation script was revised to prioritize other sub-topics such as trivia, recommendation of celebrity news about an artist, and other fresh information.

Implementation notes The Music component used multiple open Web APIs to gather the data, including Billboards, SeatGeek, Spotify, and lastFM.

The customer utterances were classified into music-specific fine grained intents (such as request for recommendation, playing songs, singing, etc). A Decision Tree classifier was trained on historical customer utterances to identify seven most popular music-related intents. We were able to achieve higher accuracy than that in Movies with more classes, as Music classes were both semantically and syntactically diverse, which allowed for cleaner division. The scoring of the responses combined both the context and Topic Classification. Keyword-based match was the primary signal for the internal music response scoring. Just as for Movies, this component also followed each response with a question, to both keep the customer engaged and to also gain additional information to improve the recommendation of the next response.

5.3 News Component

The news component is responsible for updating the customer with breaking or trending news, as well as responding to all news-related or celebrity-related queries. This component aggregates data from multiple news sources, including the provided news articles and summaries snippets from The Washington Post, Reuters newswire and RSS feeds, and a variety of other sources using the news API. For specific queries that cannot be answered with the local index above, we fall back to the Microsoft Bing news search API. We also used Google trends to identify the trending news topics for the past hour.

Information intents addressed : The news component attempts to satisfy the most common news-related information needs, specifically:

1. Trending news: list of the trending news topics.
2. Overview of the day's news (briefing or summary).
3. News from a specific category (politics, sports, technology, entertainment, business, health, world, and the U.S.)
4. News on a particular topic or entity.

Response retrieval and scoring : For ranking news on a particular topic, we retrieve the initial pool of results using the default news ranking from Bing news API. We then re-rank results by how early the topic appears in the news, because on a voice interface, it's easier for the customer to know that the response is relevant if the topic they asked for appears sooner. Also, after returning the news with the highest score, if the topic appears in multiple categories of news (for example, a story about "Facebook" could be in both business news and technology news or even in Politics), we ask the customer to pick the category they want. We also get the trending news topics for each hour using Google trends. If the customer asks for news without a particular topic, we ask them the category they want news from, in order to narrow down to their interest.

Personalization : We keep track of the users' categories of interest, and boost the news articles that belong to that category while ranking. The intuition is that for many popular news topics, we can prioritize the categories that match the customer's interests, based on the conversation so far. We also experimented with boosting entities in the conversation so far, but it is not yet clear whether that strategy is effective, since news topics can be quite diverse and entity overlap is less likely than general topic/category overlap. We specifically offer returning customers an option to hear the news on entities for which they searched in their previous conversation, but it is not clear yet whether this strategy is effective. We plan to study personalizing the experience for returning customers in more depth in the coming weeks before the end of August as described in Section 8.

5.4 Travel and Attractions Component

The Travel and Attractions component supports search and recommendation for real-time place information, through a conversation about customers' interests. Customers can either choose to talk about their preferred locations, or rely on the component's recommendations. Once customers are satisfied with the first-round interaction about popular places, the component further engages with them about their hobbies or activities, such as Hiking or Skuba Diving, and uses this information for more personalized suggestion on locations. Once the full engagement-cycle is complete, the component sends users through cross-component recommendation to a different topic.

Data and Implementation Notes The main data source is from Google Places API, which is one of the largest datasets for locations-based information. We also created a handcrafted dataset of most common locations (cities, states and nations), and this dataset is used to detect various locations from customer utterance. Detected location is used as input to Google Places API along with detected keyphrases, personalized info and location-related entities. For effective information retrieval, the component uses query-expansion technique to combine keyphrases, location entities, and any provided hobby information into a richer query. The output from Google Places API is sent to a domain-customized heuristic ranker, and uses the available metadata features to boost or reduce the retrieval scores. After the response is retrieved, the component offers follow-up suggestions of near-by or relevant places, or otherwise attempts to engage the customer in an attempt to provide even more relevant information.

5.5 Supporting Tail information needs

Video Games: component was added because topic analysis and manual log reading revealed a large amount of interest from customers. Asking about video games before the component's addition usually caused unpredictable responses, often resulting in less than satisfying conversation experience. The design of the component was to imitate a real life conversation about one of the gamers' most popular queries: upcoming games. This component finds out about the customer's console preference and saves it, so that if the customer comes back again later, the component will pick up where it left off. It is able to talk about the most popular upcoming games for the most popular gaming platforms, specifically PlayStation, Xbox, Nintendo Switch and PC. From feedback it became clear that customers may be interested in video game suggestions for their kids, and the component was extended accordingly. If this work is allowed to continue, the natural next step is to include a much more comprehensive database on video games for the next version of the component.

The 2018 FIFA World Cup: was a major sporting events during the semi-finals, and as such a temporary component was added to address the information needs around this event. The component used live scraping of multiple website data to compile summaries of whatever teams the customers wanted to hear about. In addition, it attempted to answer some basic questions about the World Cup history and other related trivia. During the tournament, we saw some interest in the World Cup from customers, which died off completely after the World Cup was over. This showed that quickly spinning up a component was possible in our IrisBot framework, and satisfied many customers.

Cars: this component is capable of identifying various car brands and talking about upcoming models of different car types. Science and technology, and specifically automotive topics, were very popular topic that people often asked. As an initial solution, we implemented a component to support this popular interest. We created a curated dataset of upcoming car models grouped by different brands, along with model descriptions and specs. Based on this dataset, this component can answer questions and recommend new cars for the interested customers.

Sports and Celebrities: While we chose not to explicitly build deep domain knowledge about sports, to support this common information need we designed a wrapper to help a customer quickly navigate to relevant News content on the sport or player or team of interest, and create an effective query to retrieve the most relevant News content. This component engages with the customer to ask for their interests such as their favorite sport or team, and then queries the News for the appropriate information, at which point the News component continues the discussion on that topic. We took a somewhat similar approach for retrieving information about celebrities: if a customer is interested in a particular well-known person who may or may not be currently in the News, we first provide a short description of that person from Wikipedia, and then suggest to the customer to search for the latest news on that person. For most celebrities or famous people, this approach is quite effective as even if that person is no longer in the news, their key accomplishments are often outlined in Wikipedia entry. Same as with sports, once the customer is directed to the News, the News component is able to continue the discussion on that person just like on any other topic by offering relevant news organized by category and related entities.

Chat, Jokes, and Stories : By the rules of the competition, we were not allowed to propose any jokes or stories, but were allowed to respond with jokes them, if customers specifically requested them. For that, we had small components, which would return a random joke/story from a curated list. As a fallback, for when classifier failed or components couldn't find a good enough response, we had a general catch-all Chat component, which was based on the A.L.I.C.E. chatbot, implemented in AIML. The original chat contained a comprehensive list of templates. We manually curated those templates to keep only the absolutely necessary ones to be able to respond with some generic answers for incomprehensible or long-winded utterances from customers, which we had no hope of responding otherwise.

5.6 Question Answering and Recommendation using Social QA archives

General QA module: Open-domain conversational search requires supporting a wide variety of information needs, including unique questions which are impossible to anticipate in advance. As a solution, we designed a general question-answering module that uses social question answering archives such as Quora.com to search for related questions to the customer's utterance, and suggest possible answers provided for that question. The Bing Web search API with extensive customization was used for this purpose. Candidate question-answer pairs are retrieved and re-ranked by the combination of semantic similarity, grammatical completeness, sentiment, and several other heuristic features. To maintain engagement, all highly-scored (and thus probably relevant and interesting to the customer) QA candidates are stored for a future recommendation on the same topic. This component was able to support obscure questions ranging from religion and philosophy to technology and house repair topics. It also required extremely aggressive filtering to avoid responding with content that violated the contest guidelines due to profanity or inappropriate topics.

Customized Animals Question-Answering Module

Based on our feedback analysis during semifinals period, we realized the need for a component that can answer questions and suggest information about animals, because many people, asked different things about various animals. Our approach was to modify the general QA module above to specifically customize it to this domain (Animals). To ensure the relevance of candidate QA pairs to the customer utterance, extra ranking features were added, such as appearance of detected animal in question, frequency of animal occurrences in responses, as well as additional semantic matching and grammatical completeness features. Lastly, to improve the relatedness of information retrieval process, handcrafted query-expansion templates were designed specifically to retrieve interesting questions about animals and to avoid offensive or shocking content to the primary audience (children): e.g., whether a particularly cute animal is delicious.

Opinion Mining and Retrieval When IrisBot is asked for its opinion on a subject, we try to return an overview of the general sentiment on that topic by using the opinions mined from the Reddit social media website. For this purpose we indexed 46 million Reddit posts from February 2018 to May

2018, along with the respective sentiment polarities (positive, negative, neutral or mixed.). If IrisBot is asked for an opinion on a topic, and if that opinion is not precomputed, we query the entity polarity index for the entity that the customer is asking about, and count the number of posts that mention that entity with positive and negative sentiments. We then return the statistics with a disclaimer that our bot does not have an opinion on this topic yet, but is aggregating what people say on the Internet on this topic. This feature was incorporated into the bot relatively late in the semi-finals period and as a result was not extensively featured in our conversation logs. Our plan is to further connect opinions to entities in the other system components to pro-actively offer opinions on movies, bands or entities if available and deemed appropriate.

6 Personalized Topic Suggestion

As we learned during this competition, one of the key aspects of a successful conversation is topic transitions and recommending the next topic of conversation. IrisBot initially attempted to recommend topics randomly, which turned out to be disastrous for cohesive conversation flow ([28], [8]). Instead, our next version pre-defined reasonable topic sequences, or scripts for conversations, and suggested topics following a predefined sequence if a customer did not express a preference or interest in any specific topic. However, we also learned that a small number of default topic sequences does not work for all customers, and instead developed a *personalized* topic suggestion module, trained on past conversations, that attempts to predict which topic a customer would be interested in discussing next. We believe this is an important innovation as it allows our bot to tailor the topic order to each customer based on a variety of traits and characteristics.

6.1 Personalized Sequential Topic Suggestion Model

To propose the next interesting topic for each customer, we introduce a Conditional Random Field (CRF)-based sequence model. To make the suggestion personalized, we investigated two approaches, outlined below. First, we introduce the CRF model for Topic suggestion, and then discuss personalization later in this section.

First, every conversation is divided into different turns, then four different feature groups were extracted from each turn. The first group is Accepted and Reject suggestions, these feature vectors have the value 1s for the accepted topics, -1s for the rejected topics and 0s for the topics that have not been proposed. This feature helps our model be more eager to pick the topics of values 1s and 0s in the future, and consequently more reluctant to suggest 1s rejected from similar conversation states. The second group is the topic classification feature, which represent the the current conversation topic. This feature could indicate the historical probability that a current state is a potential topic-switching point, or whether it should be a follow-up from the previous state. The third group is a set of contextual features, which represent previous visited states, and previous suggested topics if they had been accepted. The fourth group is a set of high level features such as the inferred gender of users [-1,1] (female, male), and friendly users [GIVENAME or NOT].

To evaluate our approach, we first experimented with the topic suggestion model in simulation mode on off-line data. To label the data, two different scenarios have been established for training and test data. For training data, if topic X was suggested in turn *i*, and a user starts talking about topic X in turn i+1, then, the label of X-ACC is assigned to turn i. If the customer rejects the suggestion and asked for something else, the label becomes X-REJ, otherwise the label is a FOLLOW-UP for one of the components. There is a small difference between labeling of the test data compared to training data, in which, if a customer in turn i, rejects the suggested topic and in turn i+n wants to talk about topic X, the the label for turn i is modified from X-REJ to X-ACC, because it ultimately matches the customer interest.

Personalized topic suggestion : We developed a Mixture of Experts model, in which each model had been trained on a specific group of customers. Two different ways to cluster customers into different groups had been attempted. The first one is clustering customers based on their past visiting frequency with the bot, resulting in two groups: returning, and new customers. The second way for clustering is based on time of day information, which is expected to correlate with customer demographics: dividing every day into into 4 different time buckets such as, morning, afternoon, evening and midnight, and then assign every customer to one of these time buckets.

For the first experiment, we used a history field in context data which keeps track of accepted and rejected topics for all customers. In this model, for customers that previously talked to the bot, a history was recorded. For feature extraction, for a customer that previously accepted rejected our suggestions n and m times on topic X respectively, a history vector like history-X [n,m] is extracted and added to original feature vector. Finally, two different models are trained on each cluster to predict suggestion topics. This model is used to predict topics for returning and new customers.

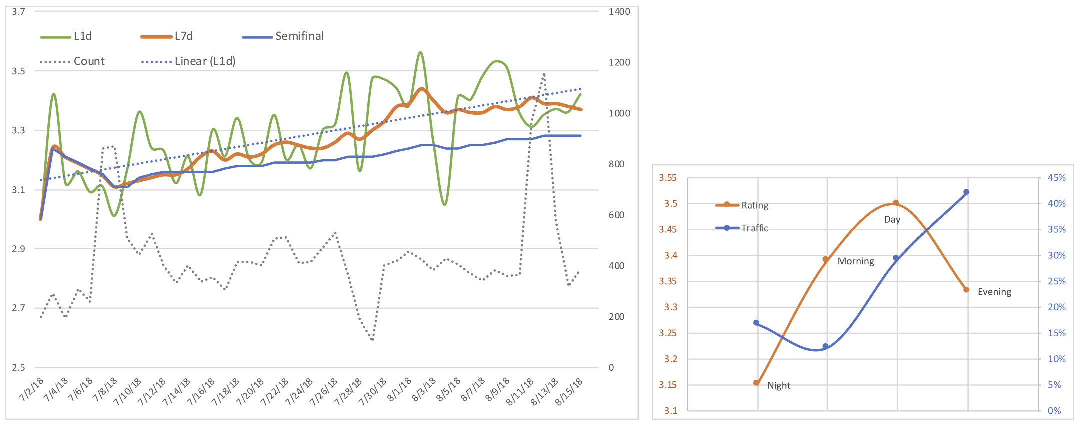
In the second experiment, 4 different models were trained on 4 different time buckets respectively, and each corresponding model was used for personalized suggestion for each customer based on the time of the day that they talk to the bot. A potential problem with this model is the time zone difference, which makes prediction less effective in our experiments described below.

6.2 Cross-Component topic suggestion

In addition to reactive topic suggestion to move the conversation forward if the customer does not express an explicit topic of interest, we proactively propose cross-component topic suggestion, whereas a component may recommend another topic as part of its response. For example, a Movie recommendation may include a famous actor, and IrisBot would suggest to the customer to look up the latest news on that actor. Another scenario we explored centers around locations: if a location is mentioned in the news or in a movie plot, that location would be recommended as a potential point of exploration for the customers and then offer them to explore the topic covered by the Travel and Attractions component. These two initial ideas are implemented in IrisBot and other cross-component connections and suggestions are being explored as future extensions of IrisBot.

7 Results and Discussion

First we report the overall performance of IrisBot and its improvements over the semi-finals period. Then, we dive into specific technical component evaluation to analyze how our proposed techniques work in isolation and together to improve the system performance. Specifically, we first report results of topic and intent classifier, which exceed the current state of the art as represented by the Amazon Annotate system (Section 7.2). Then, we report results on personalized topic suggestion (Section 7.3). We conclude this section by analyzing topic suggestion performance in live A/B testing for different groups of customers (Section 7.3), and other effects of personalization on customer behavior and satisfaction.



(a) (b)

Figure 7: Daily, weekly, and overall ratings for IrisBot during the Semi-finals period (a). Average Ratings for IrisBot for August 1-14 for estimated time intervals (b).

7.1 Overall IrisBot Performance

Overall, IrisBot improved significantly over the semifinals period. As Figure 7(a) shows, other than the dips on Aug 3 and July 28 due to system infrastructure issues, IrisBot has consistently increased customer delight and was able to provide increasingly interesting conversations for thousands of customers. To better understand customer engagement with the bot, we analyzed the topic acceptance rate of the topics we suggested. Since we expected different demographics at different times of day, we experimented with dividing our data into time buckets to come up with the order of suggestions that would be most likely to appeal to people in a given time-frame. The data in Figure 7(b) suggests that indeed there are significant differences in audience demographics in different time periods. Our current system is not able to take advantage of this information yet, but is an active area of current work on personalizing topic suggestions based on both current and past conversations (Section 6.1) but also on external factors such as time of day.

In addition, based on the summarization from the provided conversation assessment annotation data by Amazon, the quality of our responses to users utterances has been improving with respect to overall quality, coherence and engagement. As shown in Table 3, at the beginning of the semifinal period, the ratio of passable responses is about 0.53, but we achieved 0.75 close to the end. Similarly, we had noticeable improvements both on coherence and engagement, suggesting that IrisBot is evolving in the right direction, and is continuing to improve.

Period	Passable Ratio	Coherent Ratio	Engaging Ratio
06/25-07/09	0.53	0.40	0.53
07/09-07/17	0.62	0.45	0.53
07/17-07/25	0.64	0.42	0.66
07/25-08/01	0.75	0.47	0.75
08/01-08/07	0.67	0.56	0.74

Table 3: IrisBot Response Quality Summarization from Conversation Assessment Annotations Provided by Amazon.

7.2 Topic and Intent Classifier: Intrinsic Evaluation

We store user-Irisbot conversations in logs from the beginning of the contest. To evaluate the topic classifier, we manually developed a test dataset containing 1,121 utterances of different conversations from the logs, and labeled them manually. Test sample utterances were randomly selected based on the distribution of each topic in the conversation logs for users. For example, the Movie topic has about twice the number of samples as of Sports in the test set since the number of utterances about movies is twice the number of utterances about sports in the logs. The result for the classifier on 6 major topics that have intersections with the Amazon topics is reported in Table 4. The average (accuracy and Micro F1-score) for the Customized Classifier (CC), Entity Classifier (EC) and Contextual Topic Merging (CTM) on all covered topics in our bot are (0.669, 0.605), (0.676, 0.616) and (0.766, 0.719), respectively.

Topic	CC-accuracy	EC-accuracy	AA-accuracy	CTM-accuracy
Movie	0.693	0.851	0.840	0.912
Music	0.836	0.916	0.875	0.891
Sports	0.923	0.693	0.693	0.851
Celebrities	0.969	0.866	0.814	0.903
Politics	0.770	0.854	0.915	0.861
Fashion	0.823	0.95	0.76	0.851
Average	0.835	0.855	0.816	0.878

Table 4: Classification accuracy for CC: Customized Classifier, EC: Entity Classifier, AA: Amazon Annotator, CTM: Contextual Topic Merging Classifier .

To evaluate the intent classifier, the same procedure was conducted to prepare the dataset as for the topic classification. The average Micro F1 and Macro F1 scores for this dataset is 0.79 and 0.83 respectively. The detailed results for different intent classes are reported in Table 5.

Suggested Topic	Accuracy
user instruction	0.727
Opinion Request	0.980
Opinion delivery	0.560
Personal Delivery	0.880
Topic Switch	0.901
Topic acceptance	0.956
unknown	0.752
Repetition	0.941
Info Request	0.951
Dont know	0.833
Self harm	0.98
Average	**0.831**

Table 5: Intent classification accuracy.

7.3 Topic Suggestion Results

We evaluated our proposed CRF-based topic suggestion model on 3,606 different conversations from August 1 to August 15, the first 10 days of conversations are used for training, and the rest for testing. The off-line results for comparisons between the two methods Heuristic suggestion and CRF model are reported on Table. 6. The Heuristic method always suggests the next topic based on their overall popularity.

Suggested Topic	Heuristic Accuracy	CRF Accuracy
Movie	0.508	0.800
Music	0.648	0.811
Attraction	0.498	0.778
Pets-animals	0.486	0.775
news	0.397	0.796
politics	0.389	0.569
sports	0.270	0.6125
Scitech	0.417	0.639
cars	0.397	0.756
games	0.473	0.698
Average	**0.476**	**0.772**

Table 6: CRF for Topic suggestion.

If customers are clustered into returning and new cohorts, as we described above, two different models are trained on each group. As we trained a model on returning customers and predicted the suggested topic for just the returning customers, the average acceptance rate was 0.629; However, the regular model that has been trained on all users has a prediction acceptance 0.655. The results showed that, first, the returning customers are more difficult to predict, and second using specialized classifier to model just returning customers is not more effective than one general model. We also experimented with clustering customers into 4 different groups by conversation time. However, preliminary results show that using all data is still more accurate than having four different models for each special time period.

Preliminary A/B testing results: Due to lack of time we were not able to fully test the personalized predictions in the live setting. One preliminary result on 36 conversations on live data in A/B testing is inconclusive: it shows only slight improvement from 0.460 to 0.461 for returning users, which is insignificant; therefore, more data is needed to decide whether a new model is needed or a different algorithm for using the suggested predictions. We plan to explore this in our immediate current work over the next several weeks, combined with other improvements to the experience of returning customers.

7.4 News ranking and personalization

For news related to a particular topic, we get the initial pool of results from Bing news search API, and re-rank them based on how far the words in the query are from the start of the news snippet, and also based on the user's preferred news domains. By re-ranking the results based on just the distance of the entities of interest from the start, the normalized discounted cumulative gain (NDCG) improves from 0.824 to 0.895 for a set of 17 queries during the semi-finals. We calculated NDCG over the set of news results, and not *all* the documents. So in theory we are limited by how good the initial set of results is, but it is highly unlikely for a major search engine like Microsoft Bing not to have any relevant results for popular news topics.

We do not yet have enough interaction data to analyze the result of suggesting news by the category a customer has shown preference for. However, it is a useful feature especially for returning customers who are interested in news. For example, a customer may have inquired about Amazon and Apple stock and listened to news about them for several turns. Upon return, it is natural to suggest to that customer a business-related news topic. In contrast, another customer who may have previously asked for politics news and asked about what happened in Mexico, and in Iran, should be suggested world news and politics news in the future.

7.5 Effects of Personalization on conversation behavior and ratings

In order to compare the effect of personalized topic suggestion feature, we conducted a controlled experiment: identical systems with and without personalized topic suggestion feature were run side-by-side for one day on August 14th 2018. The results are promising, indicating that personalization could play a significant role in increasing customer satisfaction.

One bucket which had personalized topic suggestion received an average of 4.02 rating from 360 returning users and 3.22 rating from 2,161 new users. Another bucket which didn't have personalized topic suggestion scored the average of 3.65 rating from 178 new users, while receiving only 2.80 average rating from 52 returning users.

As a result, it is clear that treating returning customers as new, is devastating to their conversation experience. It seems natural for customers to remember their previous experience regardless of the anonymous setting of this competition. Recovering previous context, and providing personalized topic suggestions or diversifying the topics of conversation could potentially improve experience of the customers dramatically, and is a central focus of our ongoing work.

8 Conclusions and Future Work

The framework and tools we developed for IrisBot so far have shown significant promise. The state of the art topic and intent classification, flexible response ranking framework, and personalized topic suggestion are already starting to work and improve the customer experience. However, we are only scratching the surface of what is possible in conversational search and information recommendation. Many important research challenges remain, and we believe that IrisBot puts us in a perfect position to make significant headway on the following:

Response Ranking: Intent and topic classification are significantly important for detecting users' intents in conversations which highly relates to the response selection process. However, classification results could be wrong. In order to avoid bias affected by errors in intent/topic classifications, more features should be included in the response ranking function. Currently, the parametric heuristic function is used to balance the contribution of different features we selected. More experiments will be done to develop on this idea, such as integrating more features in the parametric function, and developing learning-to-rank models to obtain reasonable weights on the features.

Topic Switching: A key challenge is to decide when to switch from current conversation topic to the next one. We have experimented with topic- and intent-based switching, system-driven request (do you wish to continue talking about X), sentiment-based signals, turn-based info, and combinations of the above, which seems to work best. However, all of these could be improved with additional customer information to make personalized decisions.

Complex Agreement/Rejection Customers express complex utterances which both reject a suggestion and request something else "No, but I like Movies". We found handling this to be extremely challenging, in the end implementing both primary and secondary topic classification (primary = NO, secondary = Movies in the example above). However, many special cases exist and this problem remains unsolved. Our approach of combining contextual topic and intent classification appears promising, but more work is needed.

Cross-domain recommendation One major challenge is when combining different domains (hobbies / activities and potentially more) to the search process, it becomes challenging to retrieve and rank the results that are semantically relevant to the context. Traditional keyword-base search on metadata information works well for finding entities that contain keywords on their metadata, but performs bad on finding entities that share similar concept without any matching keywords. Incorporating 'concept relatedness' and 'contextual relevance' to the search process remains challenging.

Personalization and Experience for Returning Customers: A key challenge is how to balance familiarity (proposing topics well-received by customers in the past) and surprising and delighting returning customers by helping them discover new and interesting information. Our immediate future plans are to build on the initial personalization work we have started with IrisBot to design new experiences that combine the best of the previously explored topics and the novelty and value of proposing novel information.

In summary, our IrisBot system provides a flexible, robust, and powerful platform for experimenting with conversational search and information recommendation while maintaining an engaging and entertaining conversation with the customer.

References

[1] Sandeep Avula, Gordon Chadwick, Jaime Arguello, and Robert Capra. Searchbots: User engagement with chatbots during collaborative search. In *Proceedings of the ACM SIGIR Conference on Human Information Interaction and Retrieval, (CHIIR)*, 2018.

[2] Dan Bohus and Alexander I Rudnicky. Ravenclaw: Dialog management using hierarchical task decomposition and an expectation agenda. 2003.

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

[4] Antoine Bordes and Jason Weston. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*, 2016.

[5] Yinfei Yang Sheng-yi Kong Nan Hua Nicole Limtiaco Rhomni St John Noah Constant et al Cer, Daniel. Universal sentence encoder. In *arXiv preprint arXiv:1803.11175*, 2018.

[6] Evi Yulianti-Mark Sanderson Chen, Ruey-Cheng and W. Bruce Croft. On the benefit of incorporating external features in a neural architecture for answer sentence selection. In *n Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1017–1020. ACM, 2017.

[7] Kenneth Mark Colby, Sylvia Weber, and Franklin Dennis Hilf. Artificial paranoia. *Artificial Intelligence*, 2(1):1–25, 1971.

[8] Fei Mi Joshi, Chaitanya K. and Boi Faltings. Personalization in goal-oriented dialog. In *31st Conference on Neural Information Processing Systems (NIPS)*, 2017.

[9] Youngho Kim, Ahmed Hassan, Ryen W White, and Imed Zitouni. Modeling dwell time to predict click-level satisfaction. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 193–202. ACM, 2014.

[10] Staffan Larsson and David R Traum. Information state and dialogue management in the trindi dialogue move engine toolkit. *Natural language engineering*, 6(3-4):323–340, 2000.

[11] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[12] Oliver Lemon, Xingkun Liu, Daniel Shapiro, and Carl Tollander. Hierarchical reinforcement learning of dialogue policies in a development environment for dialogue systems: Reall-dude. In *BRANDIAL'06, Proceedings of the 10th Workshop on the Semantics and Pragmatics of Dialogue*, pages 185–186, 2006.

[13] Esther Levin, Roberto Pieraccini, and Wieland Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on speech and audio processing*, 8(1):11–23, 2000.

[14] Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*, 2017.

[15] Hien D. Nguyen and Faicel Chamroukhi. Practical and theoretical aspects of mixture-of-experts modeling: An overview. In *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, page e1246, 2018.

[16] Filip Radlinski and Nick Craswell. A theoretical framework for conversational search. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*, pages 117–126. ACM, 2017.

[17] Charles Rich and Candace L Sidner. Collagen: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction*, 8(3):315–350, 1998.

[18] Shourya Roy and L Venkata Subramaniam. Automatic generation of domain models for call centers from noisy transcriptions. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 737–744. Association for Computational Linguistics, 2006.

[19] Johanne R Trippas, Damiano Spina, Lawrence Cavedon, Hideo Joho, and Mark Sanderson. Informing the design of spoken conversational search. In *Proceedings of the ACM SIGIR Conference on Human Information Interaction and Retrieval, (CHIIR)*, 2018.

[20] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

[21] Alexandra Vtyurina, Denis Savenkov, Eugene Agichtein, and Charles LA Clarke. Exploring conversational search with humans, assistants, and wizards. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 2187–2193. ACM, 2017.

[22] Zhongyuan Wang Dawei Zhang Wang, Jin and Jun Yan. Combining knowledge with deep convolutional neural networks for short text classification. In *In Proceedings of IJCAI*, volume 350, 2017.

[23] Zihao Wang, Ali Ahmadvand, Jason Ingyu Choi, Payam Karisani, and Eugene Agichtein. Emersonbot: Information-focused conversational ai emory university at the alexa prize 2017 challenge.

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

[25] Jason D Williams. The best of both worlds: unifying conventional dialog systems and pomdps. In *INTERSPEECH*, pages 1173–1176, 2008.

[26] Jason D Williams and Steve Young. Scaling up pomdps for dialog management: The"summary pomdp"method. In *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on*, pages 177–182. IEEE, 2005.

[27] Jason D Williams and Steve Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422, 2007.

[28] Min Yang, Zhou Zhao, Wei Zhao, Xiaojun Chen, Jia Zhu, Lianqiang Zhou, and Zigang Cao. Personalized response generation via domain adaptation. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1021–1024. ACM, 2017.

[29] Junbo Zhao Zhang Xiang and Yann LeCun. Character-level convolutional networks for text classification. In *arXiv preprint arXiv:1603.03827*, 2014.