Conversational Semantic Search: Looking Beyond Web Search, Q&A and Dialog Systems

Paul A. Crook, Alex Marin, Vipul Agarwal, Samantha Anderson, Ohyoung Jang,
Aliasgar Lanewala, Karthik Tangirala, Imed Zitouni
Microsoft Corporation, Redmond, WA
[pacrook,alemari,vipulag,samande,ohjang,Aliasgar.Lanewala,kartang,izitouni]@microsoft.com

ABSTRACT

User expectations of web search are changing. They are expecting search engines to answer questions, to be more conversational, and to offer means to complete tasks on their behalf. At the same time, to increase the breadth of tasks that personal digital assistants (PDAs), such as Microsoft's Cortana or Amazon's Alexa, are capable of, PDAs need to better utilize information about the world, a significant amount of which is available in the knowledge bases and answers built for search engines. It thus seems likely that the underlying systems that power web search and PDAs will converge. This demonstration presents a system that merges elements of traditional multi-turn dialog systems with web based question answering. This demo focuses on the automatic composition of semantic functional units, *Botlets*, to generate responses to user's natural language (NL) queries. We show that such a system can be trained to combine information from search engine answers with PDA tasks to enable new user experiences.

ACM Reference Format:
Paul A. Crook, Alex Marin, Vipul Agarwal, Samantha Anderson, Ohyoung Jang, Aliasgar Lanewala, Karthik Tangirala, Imed Zitouni. 2018. Conversational Semantic Search: Looking Beyond Web Search, Q&A and Dialog Systems. In *Proceedings of WSDM 2018: The Eleventh ACM International Conference on Web Search and Data Mining , Marina Del Rey, CA, USA, February 5–9, 2018 (WSDM 2018),* 4 pages.
https://doi.org/10.1145/3159652.3160590

1 INTRODUCTION

An increasing number of companies are investing in personal digital assistants (PDAs) as gateways to capture and retain users in their ecosystem of services and devices, *e.g.* Microsoft's Cortana, Apple's Siri, Google Assistant, Amazon's Alexa, Samsung's Bixbi. While the range of skills and tasks that these PDAs can complete is rapidly increasing, their apparent intelligence is limited by the set of inhomogeneous back-end systems that are typically used to power such PDAs [6, 8]. The back-end systems typically exchange only limited information, if any, and the PDA engine as a whole often lacks an ability to connect or perform inference over the information they provide. Thus the PDA is often ignorant of the information shown by a particular back-end and is unable to connect it with subsequent turns in the conversation.

At the same time, web search engines are also evolving. Gone are the days where Google and Bing would show only blue links that users have to click on. Users are now presented with *entity cards e.g.* the sunrise definition in figure 1, or *answer cards*, *e.g.* the sunrise time answer dominating the top left of the figure 1. These cards can contain details of computed answers or the best matching well-known-entity (people, places, companies, *etc.*). Search results can also include carousels (not shown) of local restaurants or shopping opportunities related to the search query. A click, instead of leading through to a web page, can launch the user's preferred app. to complete a restaurant reservation or a purchase. This search engine richness is a significant knowledge resource that PDAs often fall-back, *e.g.* compare the left and middle panels of figure 1. Such responses are powered through both knowledge bases, and hand crafted functions. Better utilization of the rich stream of information that search engines make available could lead to a significant improvement in the usefulness and versatility of PDAs.

As search engines start to behave more like PDAs, and PDAs try to tap the knowledge built up by search engines, it seems likely that the underlying technologies will merge and a new breed of conversational-search engines will emerge. A key integration point is the semantically rich data of entities and relationships built up by search engines. With this in mind we adopt the term *Conversational Semantic Search* to broadly describe architectures such as the one presented here. We aim for an architecture that manages contextual multi-turn conversation with users. In this demo, we focus on the automatic composition of answers and PDA skills/tasks to deliver new user experience to both web search and PDA users.

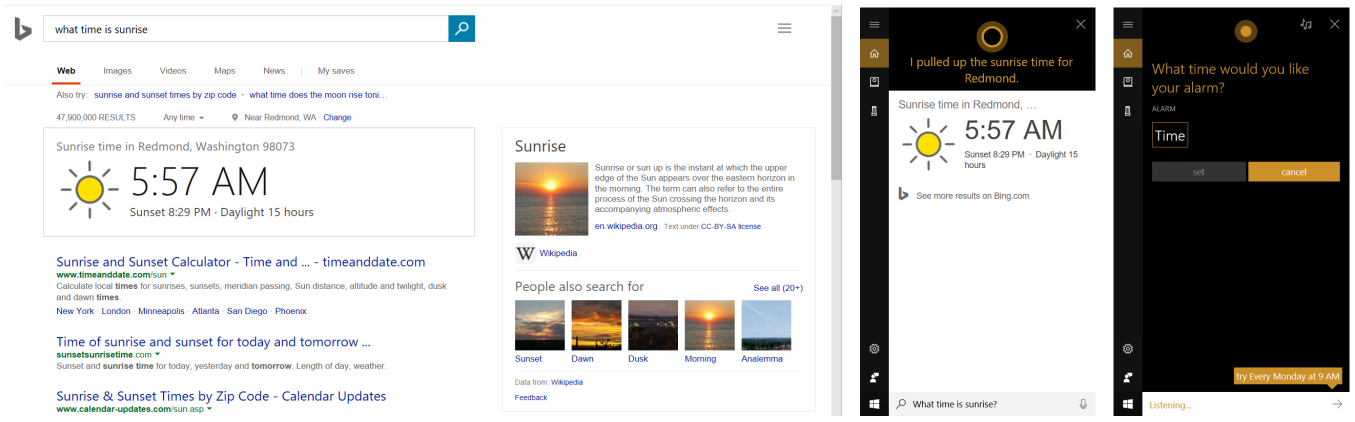
The remainder of this paper is structured as follows; section 2 briefly describes related work, section 3 describes the demonstration user experience, section 4 the system operation and architecture, and section 5 concludes.

2 BACKGROUND

The authors are not aware of prior efforts to marry semantic web search (over knowledge bases that contain both knowledge entities and actions) with conversational agents (that track state information) in a single architecture.

Dipper [2] (an implementation of an information-state dialog manager (DM) [4]), and Ravenclaw [1] are among the closest DM to the demonstrated system. In Dipper and Ravenclaw, tasks are described using sets of rules and hierarchies of agents, respectively. Each rule/agent handles a sub-part of the interaction, which is akin to the function of Botlets in the current proposal. Principal differences are that in the former systems the set of rules/agents are fixed,

WSDM 2018, February 5–9, 2018, Marina Del Rey, CA, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5581-0/18/02…$15.00
https://doi.org/10.1145/3159652.3160590



what time is sunrise

Web Images Videos Maps News | My saves

Also try: sunrise and sunset times by zip code · what time does the moon rise toni…

47,900,000 RESULTS Any time Near Redmond, WA · Change

Sunrise time in Redmond, Washington 98073

5:57 AM

Sunset 8:29 PM · Daylight 15 hours

Sunrise and Sunset Calculator - Time and … - timeanddate.com
www.timeanddate.com/sun
Calculate local **times** for sunrises, sunsets, meridian passing, Sun distance, altitude and twilight, dusk and dawn **times**.
New York · London · Minneapolis · Atlanta · San Diego · Phoenix

Time of sunrise and sunset for today and tomorrow …
sunsetsunrisetime.com
Sunset and **sunrise time** for today, yesterday and **tomorrow**. Length of day, weather.

Sunrise & Sunset Times by Zip Code - Calendar Updates
www.calendar-updates.com/sun.asp

Sunrise

Sunrise or sun up is the instant at which the upper edge of the Sun appears over the eastern horizon in the morning. The term can also refer to the entire process of the Sun crossing the horizon and its accompanying atmospheric effects.
en.wikipedia.org · Text under CC-BY-SA license

Wikipedia

People also search for See all (20+)

Sunset Dawn Dusk Morning Analemma

Data from: Wikipedia
Feedback

I pulled up the sunrise time for Redmond.

Sunrise time in Redmond, …

5:57 AM

Sunset 8:29 PM · Daylight 15 hours

See more results on Bing.com

What time is sunrise?

What time would you like your alarm?

ALARM

Time

set cancel

try Every Monday at 9 AM

Listening...

Figure 1: Left panel, Bing web search displaying an *answer card* (dominant top left) and *entity card* (right) in response to the query “what time is sunrise”. Middle panel, Cortana responding to the same spoken query. Right panel, Cortana responding to “set an alarm.”

they directly manipulate a shared dialog state, and dialog task developers predefine when rules/agents get executed. In the proposed platform, the state tracker builds and tracks the state over an evolving graph of entities and Botlets (the latter encoding “rules/agents” as mini-procedures with program counter and stack), where graph edges correspond to variables (both bound and unbound) and nodes represent data objects (entities) or Botlets. A Botlet’s availability is dynamically determined by a search step that inserts it into the graph. A trained policy decides when or whether to execute it. A motivating factor for this architecture is that it allows extension through the addition of new Botlets via mechanisms akin to web crawling and information retrieval.

Future work considers training the system using sequential decision making. In this regard, the proposed system has similarities with sequential decision formulations of interactive information retrieval and session search approaches [5, 9]. In the proposed system, user-system dialogs can be viewed as form of query refinement.

3 DEMO EXPERIENCE

Users who type “What time is sunrise” into any of today’s dominant search engines can expect their question to be immediately answered as shown in the left panel of figure 1. Similarly they can expect a PDA to understand the required information and actions that are needed in order to fulfil the request “set an alarm”. Even though that same PDA is able to answer the question “What time is sunrise?”, middle panel of figure 1, it is currently unlikely that the set-alarm dialog, right panel of figure 1, will have general access to functions which return the specific time of such events. Without access to the specific time it cannot successfully complete a dialogue like “Set an alarm”, “For when?”, “For sunrise.” Or, even more challenging in terms of the additional inference required, “Set an alarm for 20 minutes before sunrise.”

Although it is possible to engineer solutions to individual examples, for instance building the sunrise and sunset tables into a natural language understanding (NLU) component, our aim is to design a more general approach which will automatically scale to cover the possible combinations. An approach which, for example, also covers requests such as “Create a reminder to download the latest sales results an hour before my meeting with my boss.” and “Get me driving directions to my next meeting.”

Our solution is to break the problem down into functional units, which we call Botlets. Botlets are transformational functions with side-effects; mini-procedures, currently hand authored, with formal input and output semantics. They both provide mappings between semantic types and capture sub-parts of procedures required to complete specific actions on behalf of the user, including requesting information from users, or booking tickets (hence the side-effects). Botlets are invoked in sequence by a policy. Learnt execution sequences of Botlets correspond to the execution of dialogue flows required to complete tasks.

Specifically, in this demo, it is possible to ask “what time is sunrise?”, “when is 20 minutes before sunrise?”, or even "when is 20 minutes after 1 hour before sunset” (essentially some combination of qualifiers and some event) and get an answer. Additionally, it is possible to ask the system to execute a task, for example a set an alarm task, consume that answer and complete successfully, *e.g.* ask “Set an alarm for 20 minutes before sunrise.”

Internally this results in the automatic composition of Botlets to complete the required inference steps as shown in figure 3. This visualization displays a summary of the internal session state of the engine, refer to as the dynamic knowledge graph (dKG). Details of this visualization are discussed later.

4 ARCHITECTURE AND OPERATION

Figure 2 shows the system architecture. At its core is the dKG. The dKG stores the current session state which is described not only in terms of the information captured from the user, as it would be in a typical dialog system, but in addition includes Botlets and execution paths (chains of Botlets), some partially completed – see figure 3. These execution paths represent alternative analyses and hence alternate hypotheses about the user’s current question/intent and how to respond. A session can span multiple dialogs, with the system carrying forward information in the dKG between dialogs.

The orange rectangles and associated arrows form an inner update-policy-execution core that can be cycled around multiple times for a single user turn (*i.e.* a query-response pair). At the start

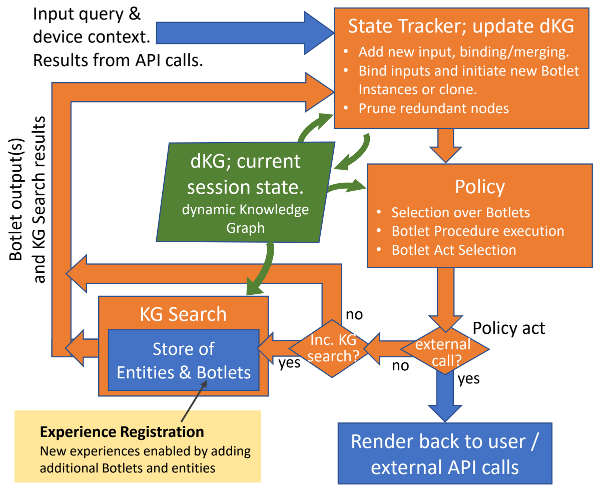

Input query &
device context.
Results from API calls.
State Tracker; update dKG
Add new input, binding/merging.
Bind inputs and initiate new Botlet Instances or clone.
Prune redundant nodes
dKG; current session state.
dynamic Knowledge Graph
Policy
Selection over Botlets
Botlet Procedure execution
Botlet Act Selection
Botlet output(s) and KG Search results
KG Search
Store of Entities & Botlets
Inc. KG search?
no
yes
external call?
Policy act
no
yes
Experience Registration
New experiences enabled by adding additional Botlets and entities
Render back to user / external API calls

Figure 2: System architecture.

of a new conversation, a user issues a query. The user query plus context from the user's device, *e.g.* GPS coordinates, local time, *etc.* is provided to the state tracker. Input processing, such as NLU, can be applied before passing the input to the state tracker or could be treated as another Botlet and executed during the update-policy-execution cycles.

The state tracker is responsible for integrating new input into the existing dkG and pruning redundant nodes. Initially, in a brand new session, the dKG would be empty with the exception of Botlets that search the knowledge graph (KG). The state tracker is also responsible for creating executable Botlet *instances* by creating copies and binding inputs to those copies. Alternative Botlet instances represent alternative hypotheses, *e.g.* in cycle 2 of figure 3, two alternative NLU services, LuBotlet and LuisBotlet, are bound to the user input string "set an alarm for 20 minutes before sunrise."

The policy box selects the action to execute in two stages. Initially, given the state of the dKG as context, it does *Botlet Selection*, *i.e.* it selects one or more from among the Botlet instances that have not yet run. The selected Botlet instances are then executed and propose acts. Those acts may consist of writing new entities into the dKG, executing an external API call, or issuing a response to the user. The policy then considers the proposed acts and selects one or more of them. Botlet instances which were initially selected but whose acts are not chosen are reset and can be reselected by Botlet Selection in subsequent cycles.

In selecting acts, the policy has to be aware that certain acts have side effects which commit the system to a particular hypothesis. We refer to such acts as *transactional*. Transactional acts include those that present information to the user, as well as API calls that purchase goods, or tickets, *etc.* Built into the policy *Act Selection* model is the assumption that it is only safe to select one transactional act at a time. For non-transactional acts multiple can be chosen.

Initially Botlet Selection can only select the built-in KG search Botlet instances. The KG contains both knowledge entities (people, places, things), as is conventionally found in knowledge bases such as Freebase, plus a collection of Botlets. Both output entities resulting from the executed Botlets and the KG search are sent to the state tracker. The state tracker integrates these results into the session as the next cycle begins.

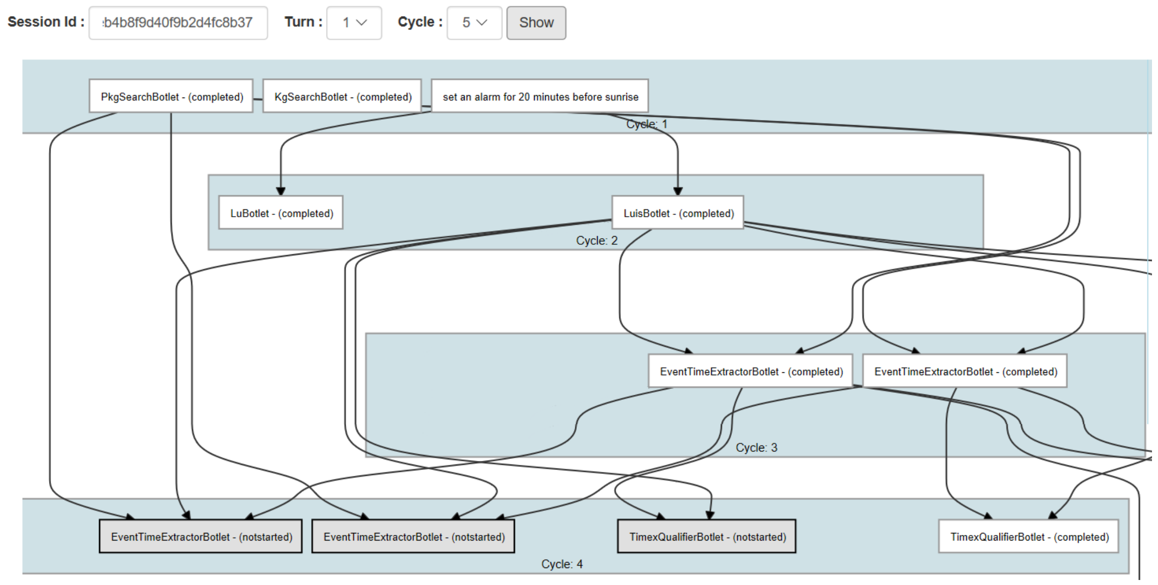
In the example session in figure 3 there are a total of 5 cycles (not all shown). During cycle 1, the user utterance is integrated into the dKG by the state tracker. Botlet Selection then selects the KG search Botlet instances (labelled PkgSearchBotlet and KgSearchBotlet). These return locations associated with the user, *e.g.* home and work, and a collection of Botlets that appear potentially useful in satisfying the user's request. In cycle 2 the state tracker binds the inputs of two of the returned Botlets creating two alternate NLU Botlet instances (labelled LuBotlet and LuisBotlet). The policy Botlet Selection selects both of them and Act Selection allows both to make external API calls to alternative NLU systems. The results are returned via the external API interface (blue input arrow) to the state tracker which in cycle 3 integrates them into the dKG; binding parts of the NLU analyses to EventTimeExtractorBotlet instances. The two EventTimeExtractorBotlet instances correspond to two hypothesis as to which sunrise event the user is referring to; today's or tomorrow's. The policy selects and accepts the proposed actions of both instances, which is to generate specific date-time entities. In cycle 4, the generated date-time entities are integrated and two TimeQualifierBotlet instances are bound. These Botlets handle the computation of time offsets, *e.g.* '20 minutes before'. Further EventTimeExtractorBotlet instances are also created. Botlet Selection selects only one of the TimeQualifierBotlet instances and accepts the action to generate a new specific date-time entity. Finally, in cycle 5, a number of SetAlarmBotlet instances are bound to the various date-time entities and Botlet Selection selects one to execute. The SetAlarmBotlet proposes as its act a response back to the user that an alarm has been set for 5:37 AM.

Multiple dialogs can be be represented in this system by multiple groups of Botlets and entities being stored in the dKG. These dialogs may be paused, either explicitly due to a user request, or implicitly if the user shifts focus to a different topic or unrelated request in the middle of a dialog. They maybe resumed in the future.

The architecture has multiple decision points where machine learning (ML) models can be deployed, such as state tracking, entity insertion/merging, Botlet instance creation and input binding, removal of redundant nodes, policy Botlet Selection, policy Act Selection, KG search. Future plans include casting the update of these models as a semi-Markov decision process (semi-MDP) [7] and re-train using Reinforcement Learning (RL) based on user feedback.

The immediate issue, however, is one of cold start. We have focused on first training a Botlet Selection model. For other components we adopted naïve rule based baselines. We also limited the shared-static-KG to a set of immediately useful entities and Botlets in order to simplify the implementation of the KG search.

The Botlet Selection model takes the current dKG as context and then emits decisions, one per Botlet instance that has not yet run, as to whether that instance should be selected in this cycle. As an initial approach, we used a sequence-encoder model as shown in figure 4. This approach allowed us to largely avoid the task of input feature engineering. Each sequence input (dkG and Botlet instance) consists of an embedding layer and gated recurrent units (GRUs) [3]. The output vectors of the two GRU sequences are concatenated and then fed to a single densely connected sigmoid output. The output is threshold at 0.5. The dKG is serialised as a linear sequence of tokens, where the maximum number of tokens used is truncated



Session Id :
:b4b8f9d40f9b2d4fc8b37
Turn :
1
Cycle :
5
Show
PkgSearchBotlet - (completed)
KgSearchBotlet - (completed)
set an alarm for 20 minutes before sunrise
Cycle: 1
LuBotlet - (completed)
LuisBotlet - (completed)
Cycle: 2
EventTimeExtractorBotlet - (completed)
EventTimeExtractorBotlet - (completed)
Cycle: 3
EventTimeExtractorBotlet - (notstarted)
EventTimeExtractorBotlet - (notstarted)
TimexQualifierBotlet - (notstarted)
TimexQualifierBotlet - (completed)
Cycle: 4

Figure 3: Screen shot from demo showing a dKG summary that illustrates information flow (arrows), *i.e.* Botlet input bindings, and selective execution. Botlets that are shown as white filled rectangles have completed execution, those shown as light grey filled rectangles have not been selected to execute by the Botlet Selection ML model.

at 12,000 retaining only the tail[1]. Botlet instances are similarly serialized to tokens but only including the instance itself and all related input entities. Maximum length of Botlet Instance tokens was limited to a few hundred.

To initially train this model we generated synthetic data by rolling out possible decision paths in response to user requests with known semantics. Each decision, from sequences that gave rise to target answers, is used a supervisory data to train the initial model. Language generation models initially trained from log data and then augment by hand generated the user requests. Using the known semantics of the requests, the exploration space for Botlet Selection could be constrained; randomized exploration within those automatically generated constraints allowed a reasonable success rate. Using this method we collected 2,860 successful single turn dialogs with an average of 6.05 cycles per dialog, and 5.87 Botlet Selection decisions per cycle, *i.e.* a total of 101,623 decisions.

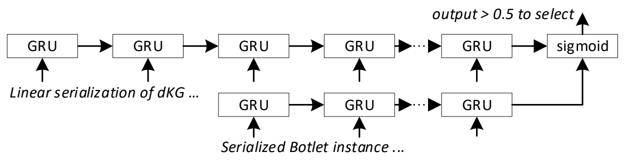
This data is then used to train the seq-2-one model. All layers of the model, including the embedding layers, were jointly trained using back-propagation.

5 CONCLUSION

We demonstrate a system that merges elements of traditional multi-turn dialog systems with web based question answering. We show that such a system can be trained to combine information from search engine answers with PDA tasks to enable new user experiences. We breakdown interpretation and processing of semantic entities, along with responding to the user, into functional units, *i.e.* Botlets. The systems is trained to automatic compose the execution of Botlets in order to complete user requests. Our approach is general enough that it can automatically scale to cover new combinations of answers and PDA task. We demonstrate it answering queries such as “What time is 20 minutes before sunrise” as well as “Set an alarm for 20 minutes before sunrise.”

Future work is focused on refining the system’s operation and accuracy. This includes training ML models for each decision point and improving those models using user feedback and RL.

[1]Max. length 62,190 tokens, 90th percentile 18,021 tokens, 75th percentile 8,295 tokens.



output > 0.5 to select
GRU
GRU
GRU
GRU
GRU
sigmoid
Linear serialization of dKG ...
GRU
GRU
GRU
Serialized Botlet instance ...

Figure 4: Seq-2-one model used for Botlet Selection.

REFERENCES

[1] D. Bohus and A. Rudnicky. 2003. RavenClaw: Dialog Management Using Hierarchical Task decomposition and an Expectation Agenda. In *Proc. Eurospeech.*

[2] J. Bos, E. Klein, O. Lemon, and T. Oka. 2003. DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture. In *In 4th SIGdial Workshop on Discourse and Dialogue.* 115–124.

[3] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR* abs/1412.3555 (2014). http://arxiv.org/abs/1412.3555

[4] S. Larsson and D. R. Traum. 2000. Information State and Dialogue Management in the TRINDI Dialogue Move Engine Toolkit. *Nat. Lang. Eng.* 6, 3-4 (Sept. 2000), 323–340.

[5] J. Luo, S. Zhang, and G. H. Yang. 2014. Win-win search: dual-agent stochastic game in session search. In *SIGIR 2014.* ACM.

[6] R. Sarikaya, P. A. Crook, A. Marin, M. Jeong, J. Robichaud, A. Çelikyilmaz, Y. Kim, A. Rochette, O. Z. Khan, X. Liu, D. Boies, T. Anastasakos, Z. Feizollahi, N. Ramesh, H. Suzuki, R. Holenstein, E. Krawczyk, and V. Radostev. 2016. An overview of end-to-end language understanding and dialog management for personal digital assistants. In *SLT.* IEEE, 391–397.

[7] R. S. Sutton, D. Precup, and S. Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112, 1 (1999), 181 – 211.

[8] Z. Wang, H. Chen, G. Wang, H. Tian, H. Wu, and H. Wang. 2014. Policy Learning for Domain Selection in an Extensible Multi-domain Spoken Dialogue System. In *Proceedings of EMNLP 2014.* Association for Computational Linguistics.

[9] Y. Zhang and C. Zhai. 2016. A sequential decision formulation of the interface card model for interactive IR. In *SIGIR 2016.* ACM, 85–94.