Developing Production-Level Conversational Interfaces with Shallow Semantic Parsing

Arushi Raghuvanshi
Cisco Systems
arushi@cisco.com

Lucien Carroll
Cisco Systems
lucienc@cisco.com

Karthik Raghunathan
Cisco Systems
ktick@cisco.com

Abstract

We demonstrate an end-to-end approach for building conversational interfaces from prototype to production that has proven to work well for a number of applications across diverse verticals. Our architecture improves on the standard domain-intent-entity classification hierarchy and dialogue management architecture by leveraging shallow semantic parsing. We observe that NLU systems for industry applications often require more structured representations of entity relations than provided by the standard hierarchy, yet without requiring full semantic parses which are often inaccurate on real-world conversational data. We distinguish two kinds of semantic properties that can be provided through shallow semantic parsing: entity groups and entity roles. We also provide live demos of conversational apps built for two different use cases: food ordering and meeting control.

1 Introduction

Conversational interfaces are a prominent feature of many consumer technology products today. Popular voice assistants, such as Alexa, Siri, Google Assistant, and Cortana, have been built using a methodical approach of domain-intent-entity classification and dialogue management that is becoming an industry standard (Dialogflow, 2018; Wit.ai, 2018; Amazon, 2018; Microsoft, 2018).

In this demo, we share best practices on developing production conversational interfaces. We provide attendees with an interactive demo which illustrates the live classification of each model in the end-to-end pipeline for different queries and use cases. The demo systems focus on two use cases in English—food ordering and meeting control—though the architecture is broadly applicable to other languages and use cases.

Furthermore, we introduce two components of the pipeline that are improvements over the industry standard—entity grouping and entity roles—which are forms of shallow semantic parsing. We demonstrate the value of these shallow semantic parses, going beyond named entity recognition without exhaustive semantic or syntactic parsing, or even full relation extraction. We first give an overview of the system architecture, then describe the shallow semantic parsing problems in more detail and compare our approach to related systems.

2 Architecture

The NLP pipeline is broken down into a series of components as illustrated in Figure 1: Intent Classifier, Entity Recognizer, Entity Resolver, Semantic Parser, Dialogue Manager, Question Answerer and Application Manager. For each component, the model type, features, and hyperparameters are tuned for the use case.

This architecture allows us to bootstrap applications with few queries per intent—tens or hundreds of queries for narrow vocabulary intents and thousands of queries for open vocabulary intents—and smoothly transition as data sets increase. Getting a deployable version of a conversational assistant on small data set sizes is key, since generating conversational data is expensive, and early in development, product decisions can often change, requiring relabeling the data. Once a model is in production, dataset sizes can grow by labeling queries from user logs. This architecture scales well and continues to give robust performance as query sizes increase by orders of magnitude. As the dataset sizes grow, the most optimal model type, features, and hyperparameters will change.

2.1 Intent Classification

A **Domain Classifier** assigns an incoming query into one of a set of pre-defined topical buckets or domains. This is a classification model that uses

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (System Demonstrations), pages 157–162
Brussels, Belgium, October 31–November 4, 2018.

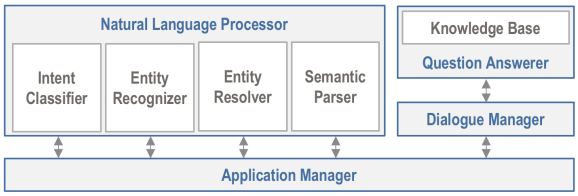


Figure 1: Natural language processing pipeline

text features like n-grams and gazetteer matches to determine which domain the vocabulary of the query is most likely in. In addition to these features that capture language and sentence structure, this component has access to the user context and the history of previous queries in the conversation. Different models such as logistic regression (LR), support vector machine (SVM), decision tree, or random forest can be selected for different use cases based on the data set size and distribution.

Intent Classifiers predict which of the domain's intents is expressed in the request. It is a text classification model like the domain classifier, but trained on data to distinguish among the types of intended dialogue acts within a domain.

Figure 2 illustrates the domain and intent classifications for the input query "I'd like a plain bagel that's warmed up and with cream cheese" for a food ordering app. In the demo, users can try tweaking the language of the query and see how it affects the model prediction. For example, when keeping the phrase "I'd like a", but changing the rest of the query to an unrelated item (e.g. "I'd like a link to a funny video"), the domain classification changes to UNRELATED. When tweaking the language from "I'd like a bagel" to a more explicit "I want to order a bagel" increases the confidence of the ORDER intent classification.

2.2 Entity Recognition

Entity Recognizers identify and label entities—the words and phrases that must be represented and interpreted to fulfill requests. It is a sequence tagging model such as a Maximum Entropy Markov Model (MEMM), Long Short-Term Memory recurrent neural network (LSTM), or a Conditional Random Field (CRF). The features depend on the model type used but can include word embeddings or gazetteer matches. The optimal model is dependent on dataset size. A MEMM model with gazetteer and n-gram features may be most robust when trained on a few hundred queries, but the LSTM model with character and

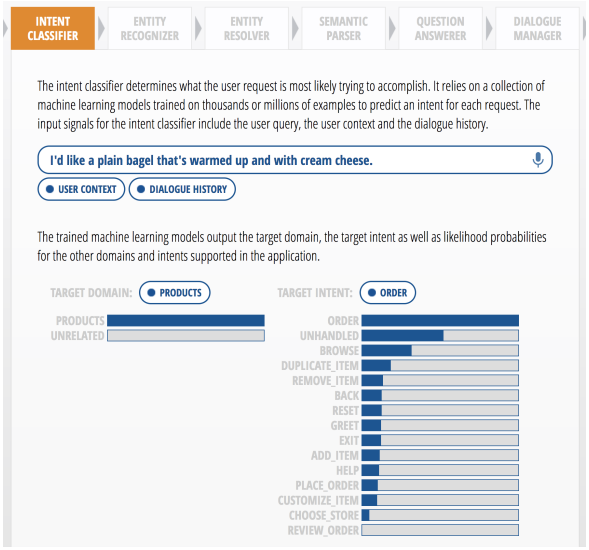


Figure 2: Domain and intent classifier demo view

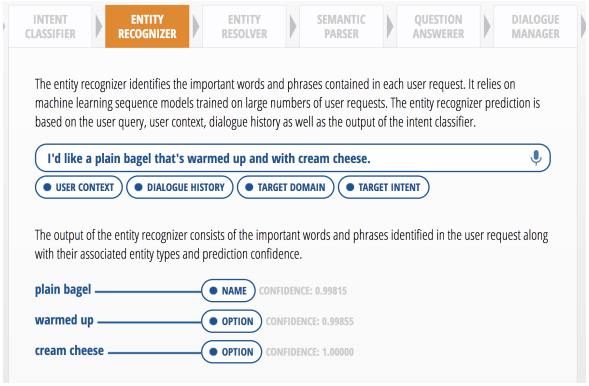


Figure 3: Entity recognizer demo view

word embeddings can have better accuracy when trained on a few hundred thousand queries.

In Figure 3 we illustrate the extracted entities from an example query and the confidence of those predictions.

2.3 Entity Resolution

An **Entity Resolver** maps each identified entity to a canonical value. For example, in the food ordering use case, the text "plain bagel" needs to be mapped to a canonical id that can be use to make an API call to place the order.

The entity resolver uses an information retrieval (IR) approach to resolve to the correct canonical form by doing text matching against an index of canonical entities. This matching uses common IR techniques of character n-gram, token n-gram, and fuzzy matching as well as tf-idf scoring. The index also contains information like synonyms for entities, so the model can correctly resolve items

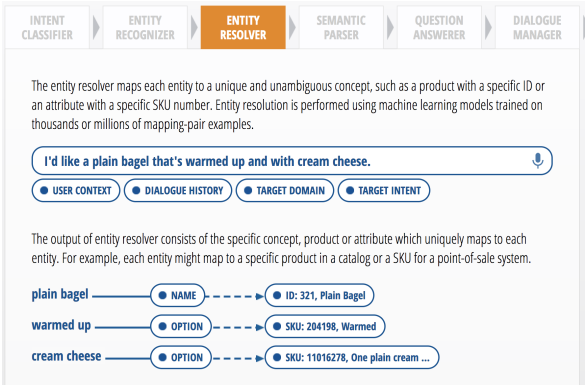


Figure 4: Entity resolver demo view

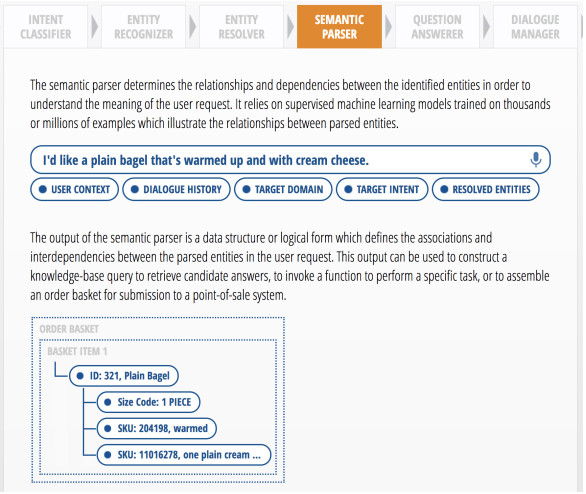


Figure 5: Semantic parser demo view

that are semantically similar but syntactically different. For example, "pasta with tomato sauce" can be resolved to "spaghetti marinara". In Figure 4, the entity text "warmed up" is resolved to the "Warmed" (SKU: 204198) as an example.

2.4 Semantic Parser

The interpretation of some queries requires a more structured representation than domains, intents and entities provide. We decompose entity relation extraction into two degenerate cases: assignment of entity relation type or **role** and association of dependent 'child' entities to a head 'parent' entity in an entity **group**. Though some use cases require both kinds of information, many use cases only require one of these, and it is productive to treat them as separable problems.

Entity Role Classifiers add another level of labeling when knowing an entity's type is not enough to interpret the entity correctly. E.g., a role label can be used to classify a numerical entity as a SIZE versus a QUANTITY. In the case "I'll have the 12 oz soda", the role of the entity "12" should be SIZE, but in the case "I'll have 3 sodas", the role of the entity "3" should be a QUANTITY.

The addition of the role classification is novel to our architecture. It is an intermediate between using only NER and doing a full semantic or syntactic parse. The benefit of doing role classification is that you can get important semantic and syntactic information in a targeted way for entities that are predefined to take multiple roles, rather than having to generate a full parse for each query. Without this learned role classification, systems often have to rely on heuristics in the dialogue manager to determine the roles of the entities. Unlike our approach, the alternative of using ruled-based heuristics does not scale to unseen language patterns.

Our role classifier can be a logistic regression, SVM, decision tree, or random forest model. The best model is selected based on the data distribution and use case. Features include n-grams, bag of words, and other entities present in the query.

A **Entity Group Parser** finds relationships between the extracted entities and groups them into a meaningful hierarchy as illustrated in Figure 5.

The language parser for entity grouping is another novel component of our architecture. A small number of configuration parameters generates a weighted context-free grammar (WCFG) over entities and the words not in entities. The developer can specify whether to allow parent-child attachment to the left or right, what the minimum or maximum number of dependent entities should be, and the preferred direction of attachment. The developer can also specify certain 'linking' words, whose occurrence between two entities increases the chance of the entities being in the specified head-dependent relationship. More details on the semantic parser are given in section 3.

2.5 Question Answerer

A **Question Answerer** queries a Knowledge Base, which encompasses all of the important world knowledge for a given application use case, to find answers to user queries. The example in Figure 6 demonstrates the ranked list of relevant results for the request "what pastries are available?".

2.6 Dialogue Manager

A **Dialogue Manager** analyzes each processed query, executes the required logic, and returns an appropriate natural language response. The lan-

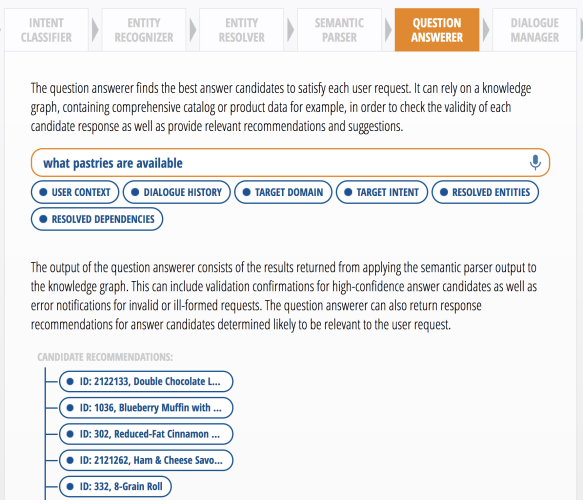


Figure 6: Question answerer demo view

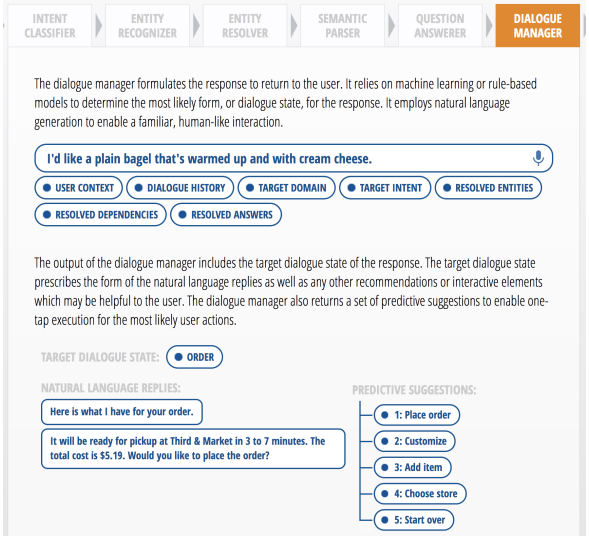


Figure 7: Dialogue manager demo view

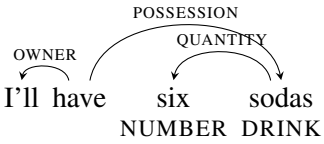
guage of the response itself is often templated, but the appropriate template is selected based on the context of the target domain, intent, entities, and results from the question answerer. In the example in Figure 7, the user's query was classified as the order intent with the correct entity and options filled in, so the dialogue manager prompts the user to confirm their order and informs them of the pickup time.

2.7 Application Manager

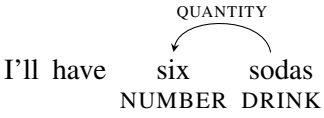
An **Application Manager** orchestrates the query workflow. It receives requests from the client including both text queries and context information, processes the requests by passing them through each of the other components, and then returns the final response to the client endpoint.

3 Shallow Semantic Parsing

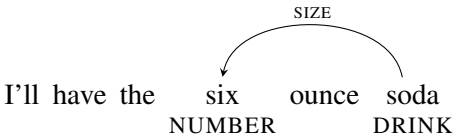
In conventional semantic parsing, a single tree of dependency relations is constructed to cover approximately a whole sentence, enabling a translation from natural language to logical form for each sentence. In the example below, the dependencies OWNER, POSSESSION and QUANTITY can be assigned within a sentence in which a NUMBER entity and DRINK entity are already recognized.



For many information retrieval tasks, including this task-oriented dialogue system, it is sufficient to just extract the relations among entities, in this case QUANTITY.



The QUANTITY relation enables the system to distinguish the sentence above from the sentence below, for example, where the same NUMBER entity has a SIZE relation.



But beyond that, it is often sufficient for the trained model to recognize the relation type, without specifying the parent. When this is the case, we can treat the relation type as a role label on the entity.

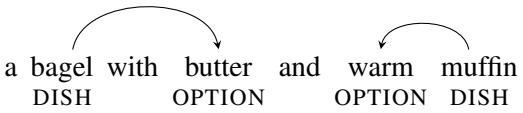
I'll have the six ounce soda
NUMBER DRINK
SIZE

I'll have six sodas
NUMBER DRINK
QUANTITY

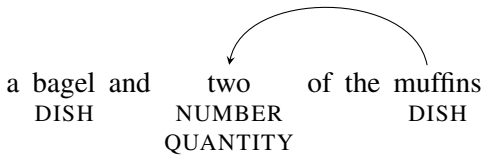
This unifies the classification problem with cases where the same disambiguation is required even though there is no explicit head, as in the query below.

I'll have six
NUMBER
QUANTITY

In contrast, entity grouping associates child entities to a parent entity without specifying a relation type, such as the OPTION entities to the corresponding DISH entities below.



Entity grouping is sufficient in cases where there is only one relation type relevant to a particular intent, or in cases where the relation type is fully determined by the associated entity types. But even when both relation type and parent-child links are required, as below, we benefit from decomposing the problem.



In the initial stages of development, when training data is limited, this allows the system to use a probabilistic model for role labeling which generalizes across cases with and without explicit heads, while using a simple deterministic parser for entity grouping. Once training data is sufficient, these can be smoothly transitioned to a fully probabilistic dependency parser, performing joint learning of entity roles and entity groups.

4 Evaluation of the Semantic Parser

To illustrate the trade-off in model approaches within this architecture as development matures, we provide an evaluation of different entity grouping systems in Table 1. The test data in this evaluation is 200 food ordering queries with more than one DISH entity, and we focus on query-level accuracy—the percentage of queries with correct labels. The baseline system Nearest Head simply assigns any dependent entity to the nearest available head entity. The Default WCFG system uses NLTK (Bird et al., 2009) to construct a CFG over entities and assigns domain-general costs to select a parse, while the Tuned WCFG uses preferences specified by the developer for this use case (e.g. particular entities tend to associate left vs. right) to adapt the cost calculation to the use case. Tuning the WCFG on 1225 queries with multiple dish entities provides an acceptable 97% accuracy. The DepP system trains the transition-based dependency parser from Spacy v. 1.9 (Honnibal and Johnson, 2015) on the entity grouping problem. With only 1225 training queries, the parser is not able to learn the structure of the problem well, sometimes leaving dependent entity types unassociated or producing impossible tag sequences. Fixup rules that force dependent entities to associate to a head entity improve the accuracy, and adding nearly 4000 training queries that only have one DISH name allows the parser to learn the basic structure better. The combination of these allows the DepP system to reach 98.5% accuracy. With smaller training data set sizes the DepP system is not viable, but once training data increases, we can expect more robust performance from the more powerful model.

System	Train #	Test %
Nearest Head	N/A	86.0
Default WCFG	N/A	91.5
Tuned WCFG	1225	97.0
DepP	1225	86.0
DepP + Fixups	1225	95.5
DepP	5106	92.0
DepP + Fixups	5106	98.5

Table 1: Evaluation of approaches to entity grouping for a food ordering use case.

5 Related Work

Our approach extends a long history of work in shallow semantics for NLU. Conceptually the most similar is work on relation extraction, as established by the Automatic Content Extraction (ACE) program (Doddington et al., 2004). For example, Culotta and Sorensen (2004) and Bunescu and Mooney (2005) use full syntactic dependency parses as features in tree-kernel SVMs to recognize relations among entities. Similarly, semantic role labeling (Màrquez et al., 2008; Palmer et al., 2010) constructs event representations by identifying arguments of predicate heads, typically treating it as a sequence labeling task. More recently, neural network models that combine sequential representations with dependency structures have obtained improvements on both relation extraction (Miwa and Bansal, 2016) and semantic role labeling (Roth and Lapata, 2016), and similar models have the power to jointly learn role labels and entity groups. However, these methods

only perform well with abundant training data, so while product decisions are in flux, working with simpler models facilitates the quick development of a production app on smaller data sets.

Some researchers have attempted full semantic parsing for task-oriented systems similar to ours. Berant et al. (2013) induce a semantic parser from question–answer pairs, and then these parses can be used to find answers for unseen questions. Chen et al. (2014) combine a semantic parser with word embeddings to induce mappings from the semantic parser's relations to the slots of a task-oriented dialogue system. In customer-facing products, systems trained by unsupervised learning have too high risk of misbehavior. Moreover, in our applied domains, either the relation head or the relation type is often predetermined or irrelevant, so that information can remain undefined or implicit, leaving a simpler problem.

6 Conclusion

We have described a dialogue system interface for an architecture which extends the traditional NLU pipeline with entity role labeling and entity grouping, forms of shallow semantic parsing. The described methodology has allowed us to consistently build production-level conversational assistants. In this demo, we hope to share these insights in an interactive way.

7 Acknowledgements

The software presented in this paper has greatly benefited from the comments and code contributions of Tim Tuttle, J.J. Jackson, Vijay Ramakrishnan, Marvin Huang, Jui-Pin Wang, Minh Tue Vo Thanh, Amrut Nagasunder, and Varsha Embar.

References

Amazon. 2018. Create the interaction model for your skill. https://developer.amazon.com/docs/custom-skills/create-the-interaction-model-for-your-skill.html.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544. Association for Computational Linguistics.

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Razvan C. Bunescu and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing - HLT '05*, pages 724–731, Morristown, NJ, USA. Association for Computational Linguistics.

Yun-Nung Chen, William Yang Wang, and Alexander I. Rudnicky. 2014. Leveraging frame semantics and distributional semantics for unsupervised semantic slot induction in spoken dialogue systems. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 584–589. IEEE.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics - ACL '04*, pages 423–es, Morristown, NJ, USA. Association for Computational Linguistics.

Dialogflow. 2018. Create and query your first agent. https://dialogflow.com/docs/getting-started/first-agent.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The Automatic Content Extraction (ACE) Program Tasks, Data, and Evaluation. *Proceedings of LREC 2004*, pages 837–840.

Matthew Honnibal and Mark Johnson. 2015. An Improved Non-monotonic Transition System for Dependency Parsing. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378.

Lluís Màrquez, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. 2008. Semantic Role Labeling: An Introduction to the Special Issue. *Computational Linguistics*, 34(2):145–159.

Microsoft. 2018. Building conversations. https://docs.microsoft.com/en-us/cortana/skills/mva32-building-conversations.

Makoto Miwa and Mohit Bansal. 2016. End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1105—-1116.

Martha Palmer, Daniel Gildea, and Nianwen Xue. 2010. Semantic Role Labeling. *Synthesis Lectures on Human Language Technologies*, 3(1):1–103.

Michael Roth and Mirella Lapata. 2016. Neural Semantic Role Labeling with Dependency Path Embeddings. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1192–1202.

Wit.ai. 2018. Build your first app. https://wit.ai/docs/quickstart.