# Alexa, Ask Wikidata! Voice interaction with knowledge graphs using Amazon Alexa

Peter Haase, Andriy Nikolov, Johannes Trame, Artem Kozlov, Daniel M. Herzig

metaphacts GmbH, Walldorf, Germany
{an, ph, jt, ak, dh}@metaphacts.com

**Abstract.** Voice-enabled user interfaces have become a popular means of interaction with various kinds of applications and services. In addition to more traditional interaction paradigms such as keyword search, voice interaction can be a convenient means of communication for many groups of users. Amazon Alexa has become a valuable tool for building custom voice-enabled applications. In this demo paper we describe how we use Amazon Alexa technologies to build a Semantic Web applications able to answer factual questions using the Wikidata knowledge graph. We describe how the Amazon Alexa voice interface allows the user to communicate with the *metaphactory* knowledge graph management platform and a reusable procedure for producing the Alexa application configuration from semantic data in an automated way.

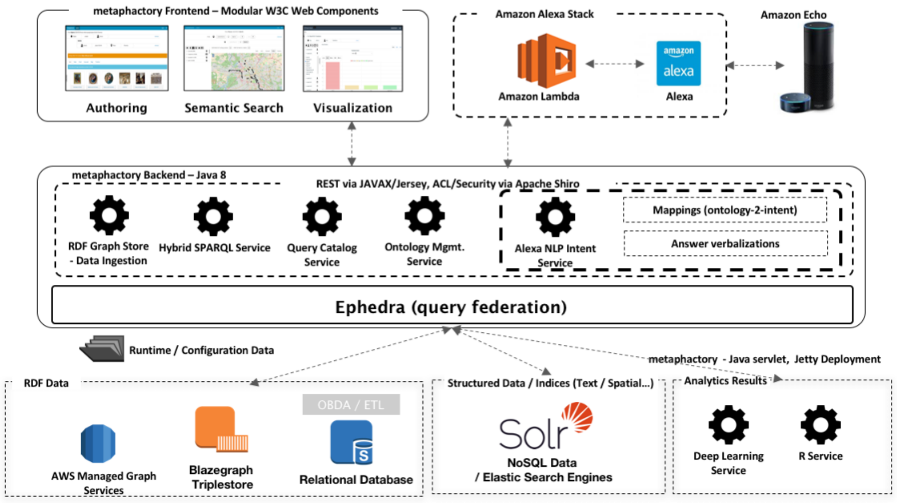
## 1 Introduction

With the advance of speech recognition technologies and the widespread use of mobile devices, voice interfaces gain popularity as an intuitive way of interacting with an application. Semantic Web applications have so far mostly used traditional paradigms such as structured queries, keyword search, and natural language queries. Voice interaction provides a valuable addition to them, providing a convenient interface for many groups of users. But the complexities of integration of voice interaction techniques made the development difficult. By separating the speech processing functionality from the application logic and making it reusable as a service, the Amazon Alexa SDK provided a flexible asset for application developers.

At metaphacts we have developed the metaphactory platform for knowledge graph management which provides a customizable application interface to semantic data stored in distributed repositories, which can be integrated using SPARQL query federation. The customized voice interface expands the platform capability by providing an alternative way of interacting with the user. In this demo paper, we describe how we used the Alexa Skills SDK to build voice-enabled Semantic Web end-user application.

## 2 Architecture

Figure 1 shows the generic architecture of the metaphactory platform. The metaphactory platform relies on SPARQL queries to access the semantic data, using RDF4J SPARQL



metaphactory Frontend – Modular W3C Web Components
Authoring
Semantic Search
Visualization
Amazon Alexa Stack
Amazon Lambda
Alexa
Amazon Echo
metaphactory Backend – Java 8
REST via JAVAX/Jersey, ACL/Security via Apache Shiro
RDF Graph Store - Data Ingestion
Hybrid SPARQL Service
Query Catalog Service
Ontology Mgmt. Service
Alexa NLP Intent Service
Mappings (ontology-2-intent)
Answer verbalizations
Ephedra (query federation)
Runtime / Configuration Data
metaphactory - Java servlet, Jetty Deployment
RDF Data
AWS Managed Graph Services
Blazegraph Triplestore
OBDA / ETL
Relational Database
Structured Data / Indices (Text / Spatial...)
Solr
NoSQL Data / Elastic Search Engines
Analytics Results
Deep Learning Service
R Service

Fig. 1. The metaphactory platform architecture.

API. The platforms supports both a single-repository scenario as well a virtual integration scenario: the data from multiple data sources are accessed using a federated SPARQL query processing engine Ephedra. On top of these integrated semantic data, the platform provides a set of services (e.g., SPARQL interface, label service, etc.), which in turn are utilized by the frontend layer. The user interface is based on a customizable templating mechanism which provides various UI elements for data visualization, authoring, and search configurable as HTML5 components.

In this demo, we present an alternative interaction mechanism we developed to be used alongside these more traditional paradigms: the voice interaction using natural language questions. The Amazon Alexa framework allows the developer to define a specific service (called *Skill*) which will then process user requests targeted at it. To enable accessing semantic data using the skill framework, we developed a mechanism to generate the definition of a skill automatically from a given knowledge graph containing factual information. As a demonstration example, we defined a skill that realizes semantic search over Wikidata and allows the user to query about specific facts by asking questions like “Alexa, ask Wikidata who is the mayor of Paris”. In the following section we describe how an Alexa skill is realized and how we can bootstrap a skill model from the underlying structured data.

## 3 Generating Alexa Intent Models for Knowledge Graphs

The Alexa voice interaction mechanism involves two functional parts: an *Alexa skill* which provides an abstraction over complex voice processing and generation services, and an Amazon *Lambda service* which realizes the application logic.

An Alexa skill is responsible for processing the voice messages from the user and transforming them into specific service requests with provided parameters. Each skill can define a number of request types called *intents*. One intent represents a call to a certain service and can be mapped to several *utterances* (natural language phrases). An utterance can refer to one or more *slots* (request parameters). When receiving a user's question "Alexa, ask Wikidata who is the mayor of Paris", the Alexa service extracts the skill name (Wikidata), identifies the intent corresponding to the question "who is the mayor of…?" and the corresponding Lambda service, and invokes this Lambda service passing as parameters the ID of the intent (e.g., "headofgovernment") and the request parameter ("Paris"). The Lambda service, in turn, contains the application logic responsible for processing the request and returning back the verbalization of the answer.

To realize a voice interface for the semantic data, our method performs a mapping from an intent to a SPARQL query pattern. In our demonstration scenario, we used the Wikidata repository, although the approach is generic and could be used with a different repository or a combination of several ones using the Ephedra federation engine. Wikidata represents a factual knowledge base where facts about entities are represented using direct properties. Thus, a specific type of information need is determined by the corresponding ontological property. To answer our example question, we need to find the value of the property *P6* "head of government" for the entity *Q90* "Paris".

To handle such a direct factual question, our Lambda function uses a SPARQL query pattern of the following form:

```
SELECT ?answer WHERE {
    ?uri <http://www.w3.org/2000/01/rdf-schema#label> ?label.
    ?label <http://www.bigdata.com/rdf/search#search> "${entity}".
    ?label <http://www.bigdata.com/rdf/search#minRelevance> "0.5".
    ?label <http://www.bigdata.com/rdf/search#matchAllTerms> "true".
    ?uri ${property} ?answer .
} LIMIT 1
```

Given a question about an entity (expressed as a string "Paris"), this query first disambiguates it using the full-text search capability of the Wikidata backend[1] and then retrieves the value of the given property. We use an automated procedure to bootstrap the Alexa skill definition and generate descriptions of intents as well as example entities. Each property in the ontology, which we want to support, is translated into a single intent (for simple factual questions all intents can map to the SPARQL query pattern shown above). For this intent, we generate possible utterances (sample questions) using metadata for the property. Property labels (*skos:prefLabel*, *skos:altLabel*) are used to generate possible question topics: e.g., "head of government of {entity}" and "mayor of {entity}". Ranges of the property are used to determine suitable question words: e.g., for the property *P6* which has instances of the class *Q5* "human", the system uses the question word "who": "Who is the head of government of {entity}?" and "Who is the mayor of {entity}?". For each intent description generated in this way, we use the same question topic to generate the answer verbalization pattern: i.e., "The head of government of Paris is {answer}". Alternative verbalization patterns are used for query results which can return multiple values: e.g., "Who is the child of Barack Obama" will be answered with a phrase "Your question about the child of Barack Obama has two answers: Malia Obama and Natasha Obama". The mappings between the intents, properties, SPARQL templates, and answer verbalizations are stored in the platform and provided by the Alexa NLP Intent Service at the platform backend.

[1] http://www.blazegraph.com/

The second necessary component of the skill model is the list of example slot values: in our case, possible words and phrases which can appear as values for the input parameter "{entity}". The examples are used by Alexa as training data to recognize correctly the values pronounced by the user. To generate these examples, we selected the most popular instances of Wikidata using their PageRank score (Alexa has a limit of 1000 possible training examples). We then select the representative properties (those with the highest number of statements and applicable to popular entities) to include into the skill model to fit into the Alexa limit of 125 single-slot intents.

## 4 Demonstration

Our demonstration is available in the Alexa Skill Store under the name "metaphacts" [2]. It can be tested using any Amazon Alexa-supporting device. The demo can answer questions like:

- "Alexa, ask Wikidata what is the currency of Cameroon?"
- "Alexa, ask Wikidata who is the wife of Bill Gates?"
- "Alexa, ask Wikidata when was the birthdate of Albert Einstein?"
- "Alexa, ask Wikidata to describe Angela Merkel"

The answers are returned both using the voice output and cards which can be viewed using the Alexa mobile app.

## 5 Conclusion and Outlook

In this demo, we show a reusable approach for generating Alexa voice interface over a semantic repository. The approach includes a procedure for producing mappings between Alexa configuration concepts (intents, utterances, verbalizations) and SPARQL query patterns and a reusable processing module implemented as an Amazon Lambda function, which utilizes these mappings to answer the user's questions.

We are planning to extend it to support more expressive patterns for different types of questions (e.g., comparison and aggregation). Another direction involves adding support for multi-linguality by utilizing the language-specific labels of Wikidata entities and Alexa support for German language. Finally, to be able to support voice interactions in commercial Semantic Web applications, we are working on the reusable ontological model for expressing Alexa configuration metadata as well as on the user interface for manual engineering of this information.

**Acknowledgements**

This work has been supported by the Eurostars project DIESEL (E!9367).

[2] https://www.amazon.com/metaphacts/dp/B0745KLCFX