DEVICE/CLOUD COLLABORATION FRAMEWORK FOR INTELLIGENCE APPLICATIONS

Prepared By,
Shelly Shiju George
Assistant Professor

INTRODUCTION

- Cloud computing is now an established computing paradigm that offers on-demand computing and storage resources to the users who cannot afford the expenditure on computing equipment and management workforce.
- This computing paradigm first led to the notable commercial success of Amazon's EC2 and Microsoft's Azure.
- These companies have adopted the business model of renting out their virtualized resources to the public.
- More recently, Google and Facebook are now utilizing their data-center-based clouds to internally run machine-learning algorithms based on the large volume of data collected from their users.
- Google runs a popular proactive service, Google Now, which gives individualized recommendations based on the user's context inferred from the personal data.
- Facebook leverages the user-uploaded images and social-network data to automatically recognize users and the relationship among them.

BACKGROUND AND RELATED WORK

- We present a novel cloud-computing framework that improves both the scalability and the privacy-protection mechanism.
- At a high level, this framework leverages the compute and storage resources on the smart mobile devices.
- Also, this framework enables security solutions that protect privacy without degrading the quality of applications.
- Note that we focus on the applications that offer personalized intelligence service.
- Therefore, we demonstrate how the selected real-world intelligence applications take advantage of the new cloud-computing framework.

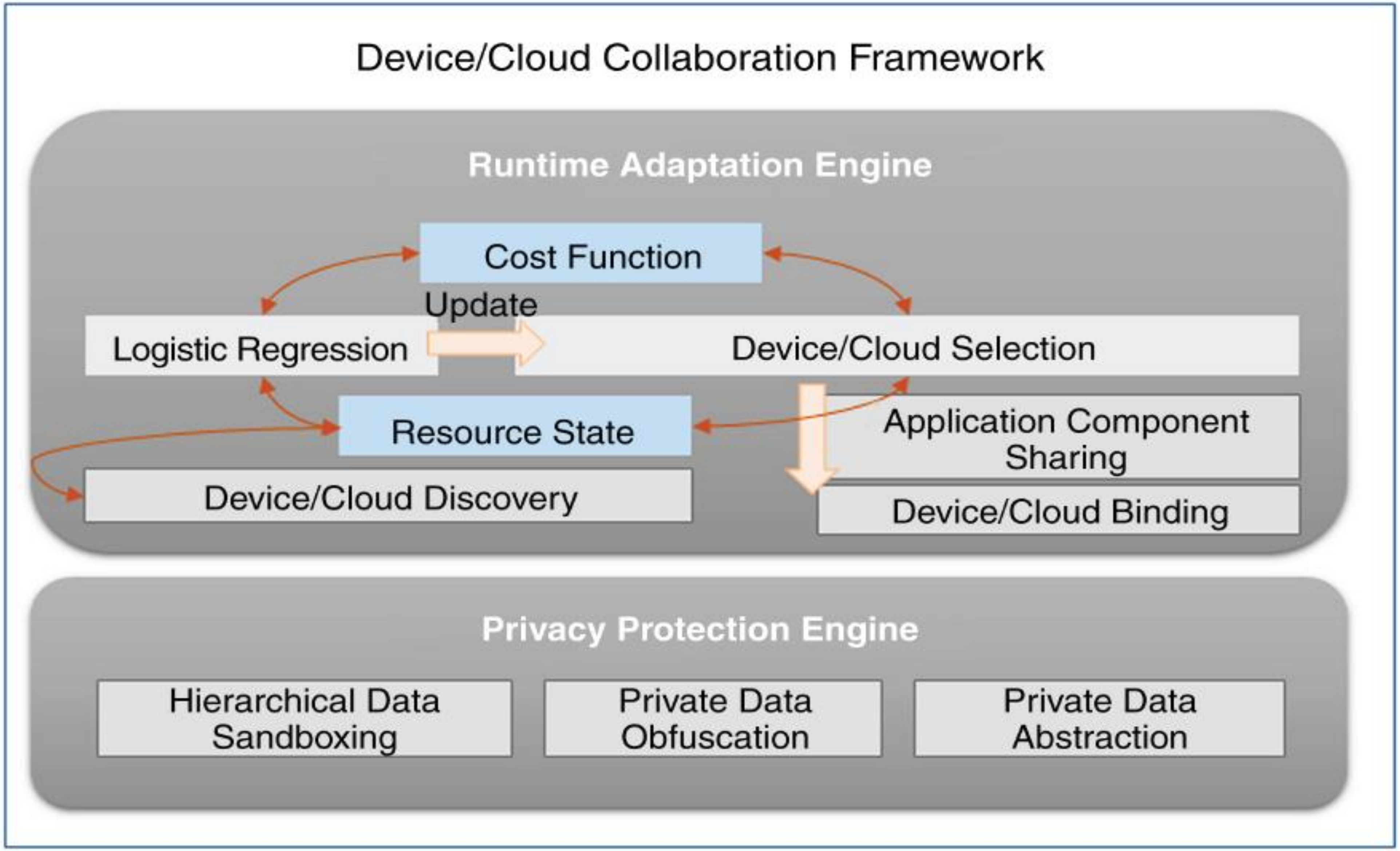
DEVICE/CLOUD COLLABORATION FRAMEWORK

- POWERFUL SMART MOBILE DEVICES
- RUNTIME ADAPTATION ENGINE
- PRIVACY-PROTECTION SOLUTION

POWERFUL SMART MOBILE DEVICES

- Smartphones nowadays have enough computing capacity to process various computing tasks.
- However, the device usage can be constrained by limited battery life and network connectivity.
- Therefore, we can consider utilizing the highly available cloud resources in addition to the device.
- In following Figure, a high-level layout of our device/cloud collaboration framework is illustrated.

High-Level Layout of the Device/Cloud Collaboration Framework



RUNTIME ADAPTATION ENGINE

- We are given a task of processing a query issued over voice, and assume that we have a lightweight mobile version of a voice-query processing engine that is embedded in a smartphone.
- This mobile engine can answer the given query without the cost of transferring the voice data to the cloud over the network.
- However, it will consume the limited battery life of the device, and/or the accuracy of the result may not necessarily be as good as that of the cloud-based query processing engine, which runs on resources with higher compute capacity.
- If the lightweight voice-query engine returns a poor result, then the user may have to issue the query redundantly to the cloud-based query processing engine with the hope of getting a better result.
- This may hurt the overall quality of experience (QoE).
- This calls for a decision mechanism that automatically selects a better agent that can execute a given task.
- With the automatic selection process in place, users do not have to worry about going through extra interaction cycles for determining where to run a job.

- The **<u>Runtime Adaptation Engine (RAE)</u>** sits at the core of our framework, as shown in figure.
- The RAE maintains a list of available devices and cloud to utilize Device/Cloud Discovery, and monitors the state of their available resources.
- The RAE employs a **<u>Logistic Regression algorithm</u>** to learn the most cost-effective policy for distributing tasks among devices and cloud, given the resource state.
- Here, the definition of the **<u>cost function is the weighted sum of the resource state (such as battery life), network, and CPU usage</u>**.
- The policy obtained by running the Logistic Regression is enforced by **<u>the Device/Cloud Selection module</u>** that chooses the most economical compute resources, based on the expected cost-value for a given task.

- The mechanism of the RAE is actually an **autonomous agent**, which can be deployed on each device and cloud.
- RAEs communicate with each other to transparently share the resource state for determining the ways to distribute a given workload.
- **The cloud-side RAE can also model its own cost function** as a weighted sum of residual CPU cycles and storage space across the entire infrastructure.
- If the **residual capacity falls below particular thresholds, the cloud may have to reject the resource-sharing request** coming from the paired devices.
- This is because **running the requested task would be too costly**.
- Specifically, the **cloud-side RAE advises the device-side RAE to either execute the task within the device or simply wait for the compute resource** on the cloud to be freed up.

- The RAEs on the devices and the cloud make decisions autonomously, without any supporting brokerage system in the middle.
- However, **the device-side RAE has the burden of periodically monitoring the state of the cloud resources**.
- On the other hand, **the cloud-side RAE does not have to monitor the resource state of the millions of paired devices**, as the cloud makes a relatively simple decision, that is, to either reject or accept a task-execution request.
- By default, our framework does not consider offloading cloud-initiated tasks to the devices.
- Suppose a device (device-A) wants to collaborate with its neighboring device (device-B), which is not equipped with appropriate application components to process a given task.
- Then, device-A can transfer the necessary application components to device-B through the **Application Component Sharing**.

PRIVACY-PROTECTION SOLUTION

- We turn our attention to the privacy-protection problem.
- Suppose we want to provide a location-based service to the users.
- To provide this service, location data such as the visited GPS coordinates, the **point of interest (POI)**, and the time of visit have to be collected first.
- Based on these location data, the mobility pattern and the interest of individual users can be inferred.
- However, these data contain personal information. Hence, for these data, we can ask the user to decide whether to transfer them to cloud or not when accepting the location-based service.
- Based on the decision made by the user, our framework can assess the expected service-quality for the users.

- Our framework employs the technology of protecting privacy by sandboxing hierarchically organized application-data.
- This technology, implemented in the Hierarchical Data Sandboxing module, **<u>supports the user to explicitly specify a group of data in the hierarchy to be shared with the cloud or not.</u>**
- The group of data that is set to be kept only within a device will be protected by **<u>sandboxing</u>**.
- Although this approach supports a specification of fine-grained privacy-protection policies, such a declarative approach would be too cumbersome for many typical users.

- Thus, we can seek an **<u>alternative solution of obfuscating</u>** (encrypting) data to be transferred to the neighboring devices or to the cloud (Data Obfuscation in Figure).
- A natural solution is to encrypt the data upon transferring to cloud.
- However, the encrypted data can be revealed through decryption-key theft from the compromised servers on the cloud or by spoofing on the tapped network.
- For these security threats, the cost of countermeasures, such as secret (eg, decryption keys) sharing across replicated servers, is nonneligible.
- Instead, we can have a lighter approach of letting the cloud analyze the obfuscated data without deciphering it, and letting the device revert the obfuscated part of the analysis result generated at the cloud side.

- For instance, suppose a user wants to receive a location-based recommendation based on the personal log of visited POIs.
- Both the POI itself and the time of POI visits are first obfuscated on the device side.
- The mapping between the original data and the encrypted data is kept on the device side.
- The device sends over the encrypted data to the cloud that does not have a decryption key to decipher the encrypted data.
- On the cloud side, data analytics such as casual reasoning through sequence mining are conducted, based on the encrypted information (eg, POIs and time of visits).

APPLICATIONS OF DEVICE/CLOUD COLLABORATION

The selected applications offer the following functionalities:

– context-aware proactive suggestion,

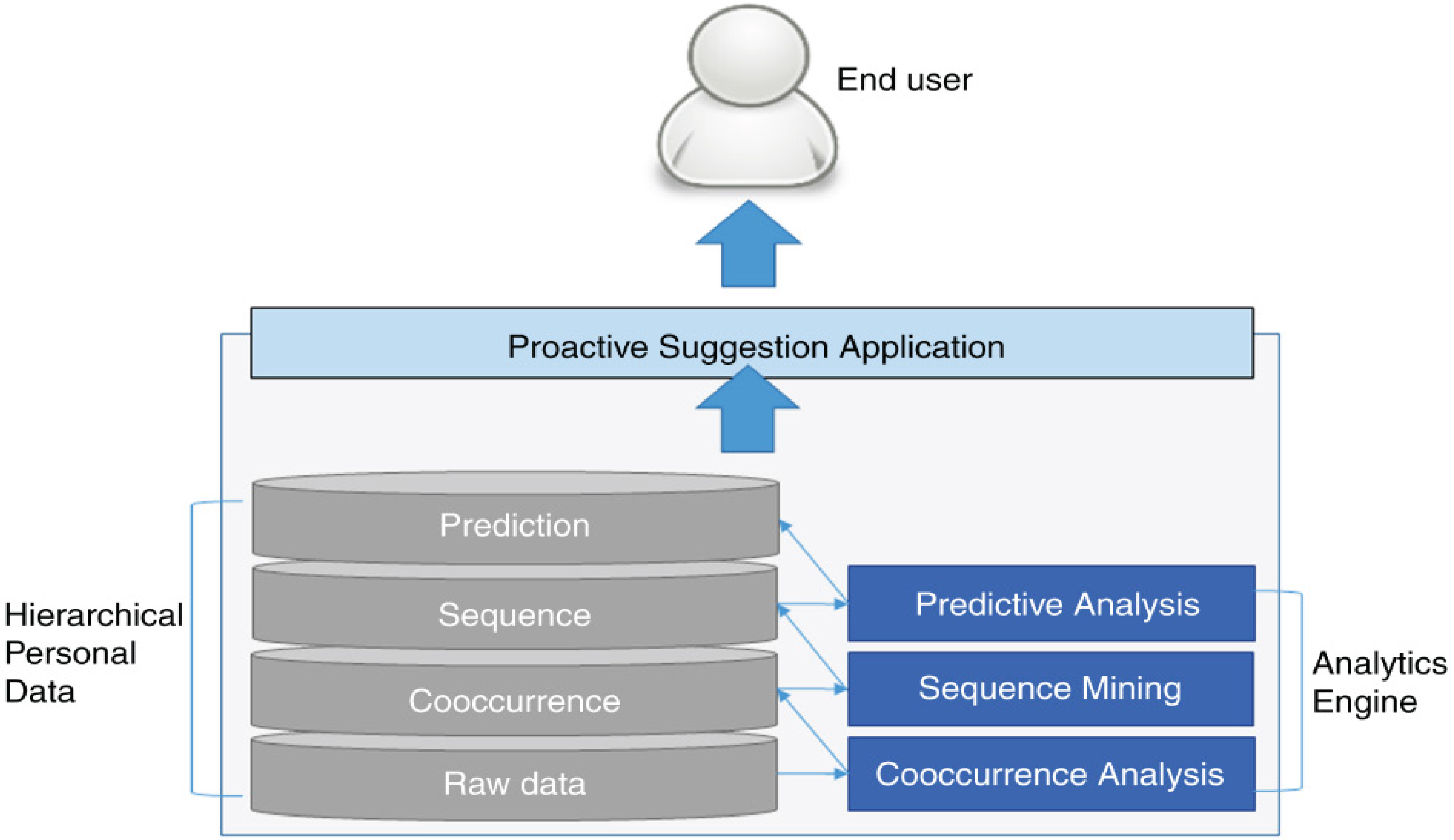
– semantic QA caching, and

– automatic image/speech recognition.

CONTEXT-AWARE PROACTIVE SUGGESTION

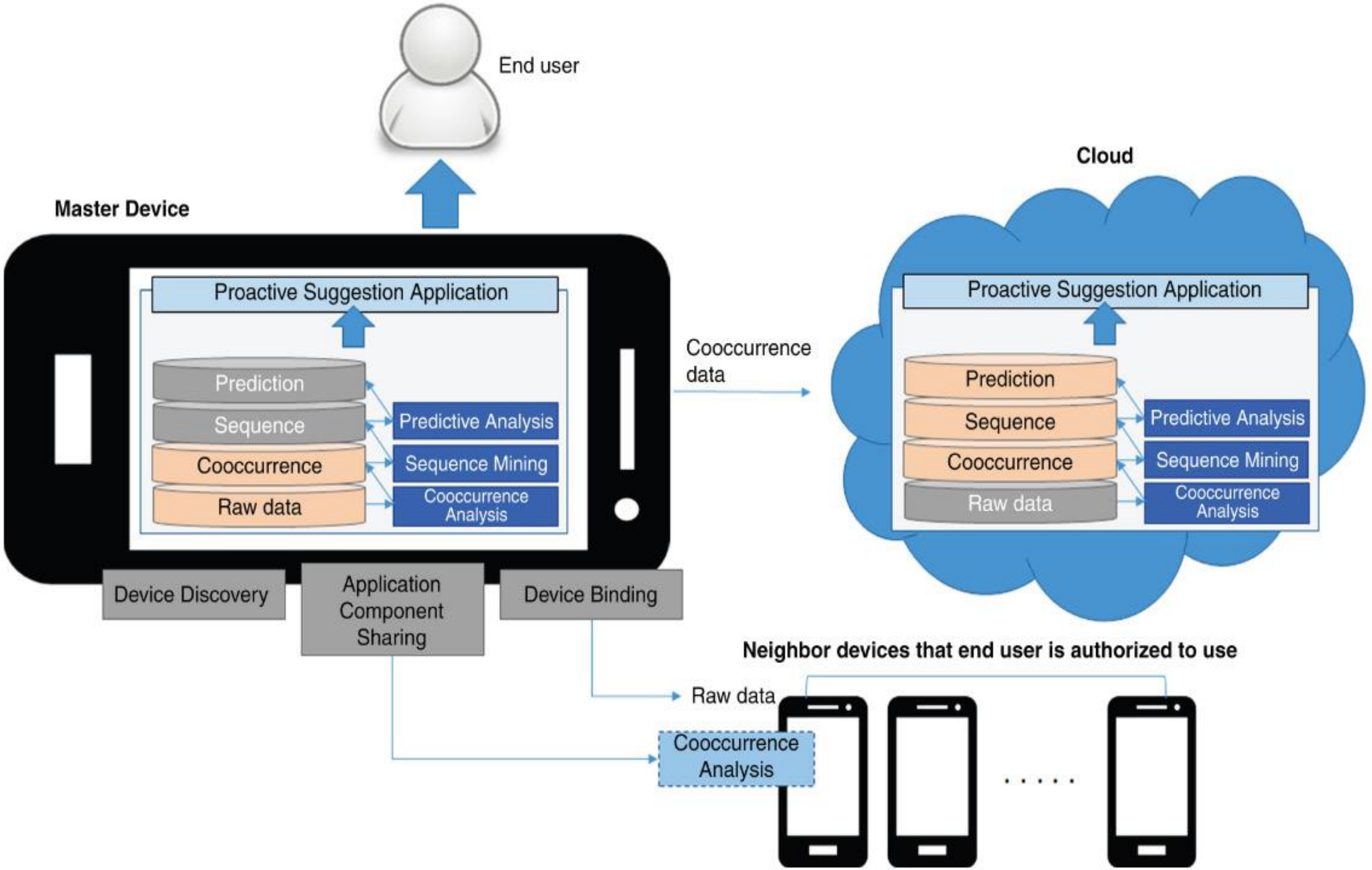
- Based on the personal data collected on each mobile device, we have devised **Proactive Suggestion (PS),** an application that makes context-aware recommendations.
- Analytics engines of PS produce hierarchical personal data that are interdependent to each other.
- Raw data such as GPS coordinates, call logs, application usage, and search queries are fed to a Cooccurrence Analysis engine, which is responsible for identifying activities that occurred at the same time.

High-Level Layout of the Core Components for the Proactive Suggestion Application



- The cooccurrence analysis engine may recognize that a user listens to live streaming music while walking in the park.
- Given such cooccurrence data, the Sequence Mining engine can infer causal relationships between personal activities that occurred over time.
- The recognized sequential patterns can be fed into the Predictive Analysis engine to assess the probability of a particular activity taking place in a certain context.

An Example of Utilizing the Device-Collaboration Framework for the Proactive Suggestion Application



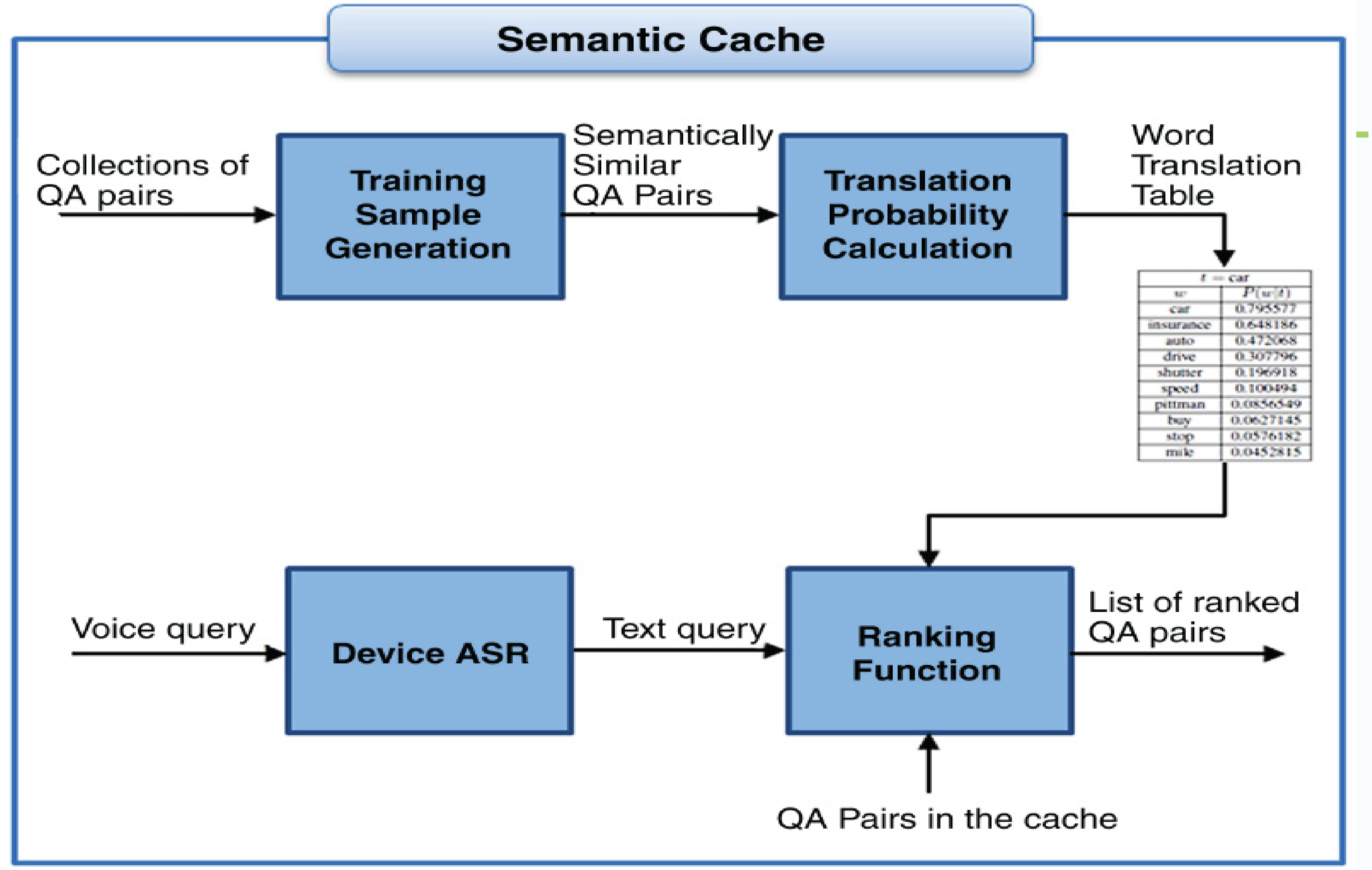
- The above Figure illustrates how PS implements the device/cloud collaboration framework.
- The master device can discover neighboring devices that the end user is authorized to use (Device Discovery).
- The master device can send over the data to one of the neighboring devices that has sufficient compute capacity (Device Binding).
- The neighboring device can retrieve an appropriate analytics engine for processing the data sent by the master device (Application Component Sharing).
- In this example, the highlighted pieces of data on the master device are shared between cloud and neighboring devices.

- The PS application initially opted for the Hierarchical Data Sandboxing for an explicit and declarative privacy-protection method.
- We could not afford to run an alternative privacy-protection method based on the data obfuscation, due to the limited resources on the device that was already bogged down by the analytics work.
- However, recall that our framework is flexible enough to allow user-defined cost functions.
- For example, if the cost of running an analytics operation (eg, the cost of consuming battery life) is excessive, then the Device/Cloud Selection module in the framework may decide to transfer the analytics task to the cloud or simply wait for the battery level to rise above the configured thresholds.
- It turned out that transferring the data over the network consumed as much energy as running the analytics operation within the device.
- Thus, the Device/Cloud Selection module opted for waiting until the battery got charged above the configured level.

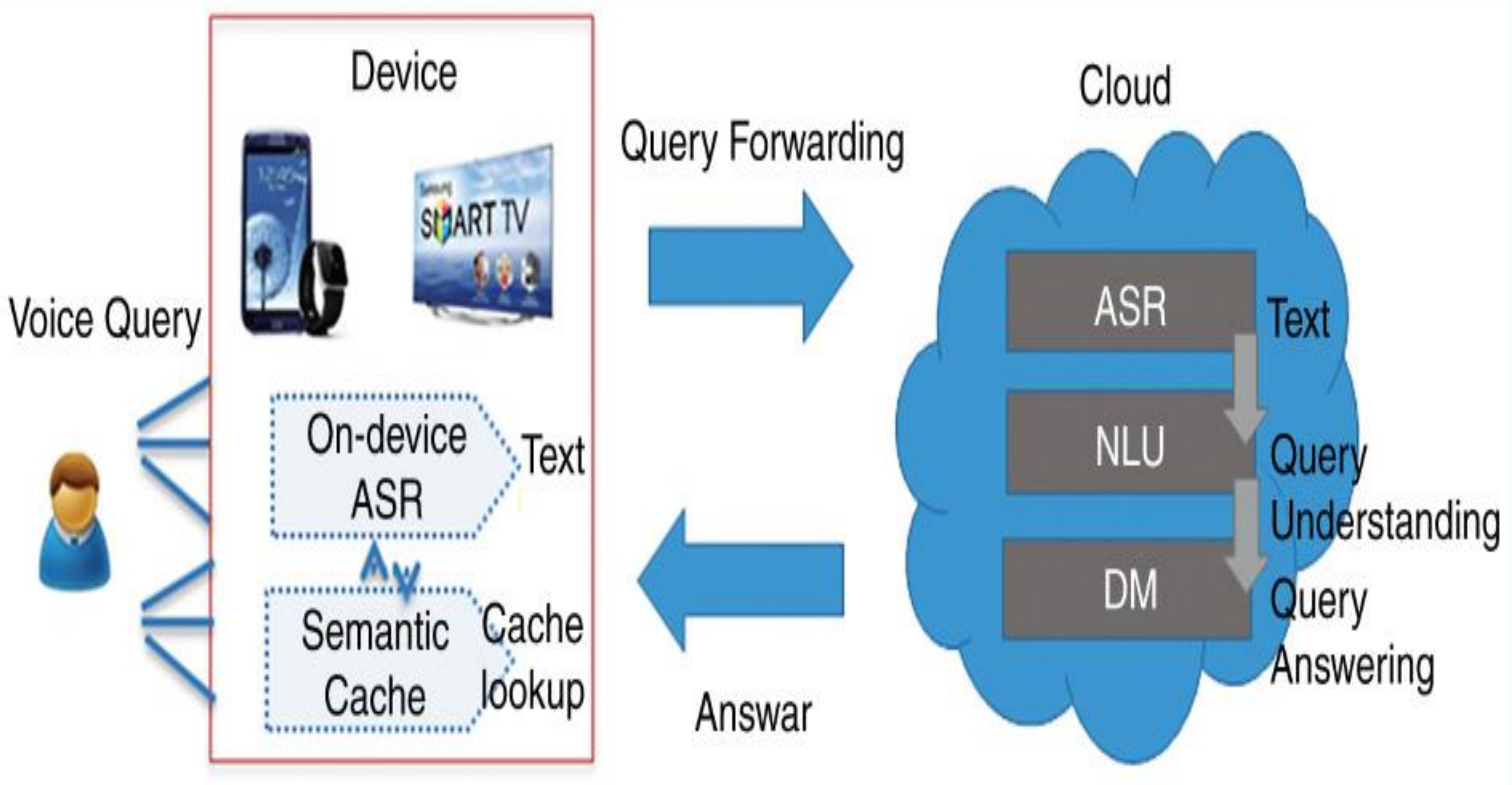
SEMANTIC QA CACHE

- Semantic QA cache is a mobile application that retrieves answers to a given query from the cache filled with answers to the semantically similar queries issued in the past.
- Semantic QA cache can be useful when there is no Internet connectivity or when the user is not in favor of transferring private queries to the cloud.
- The following Figure illustrates how the semantic QA cache is managed. Semantic QA cache returns a list of similar queries and the associated answers.
- Semantic QA cache constantly updates ranking function based on the word-translation table as explained in.
- The ranking function measures the similarity between a newly issued query and the queries measured in the past.

Illustrations of the Technique to Cluster Semantically Similar QA Pairs for Retrieving an Answer for a Newly Given Query Without Asking the QA Engine on the Cloud Side



Semantic QA Cache Implementing the Device/Cloud Collaboration Framework



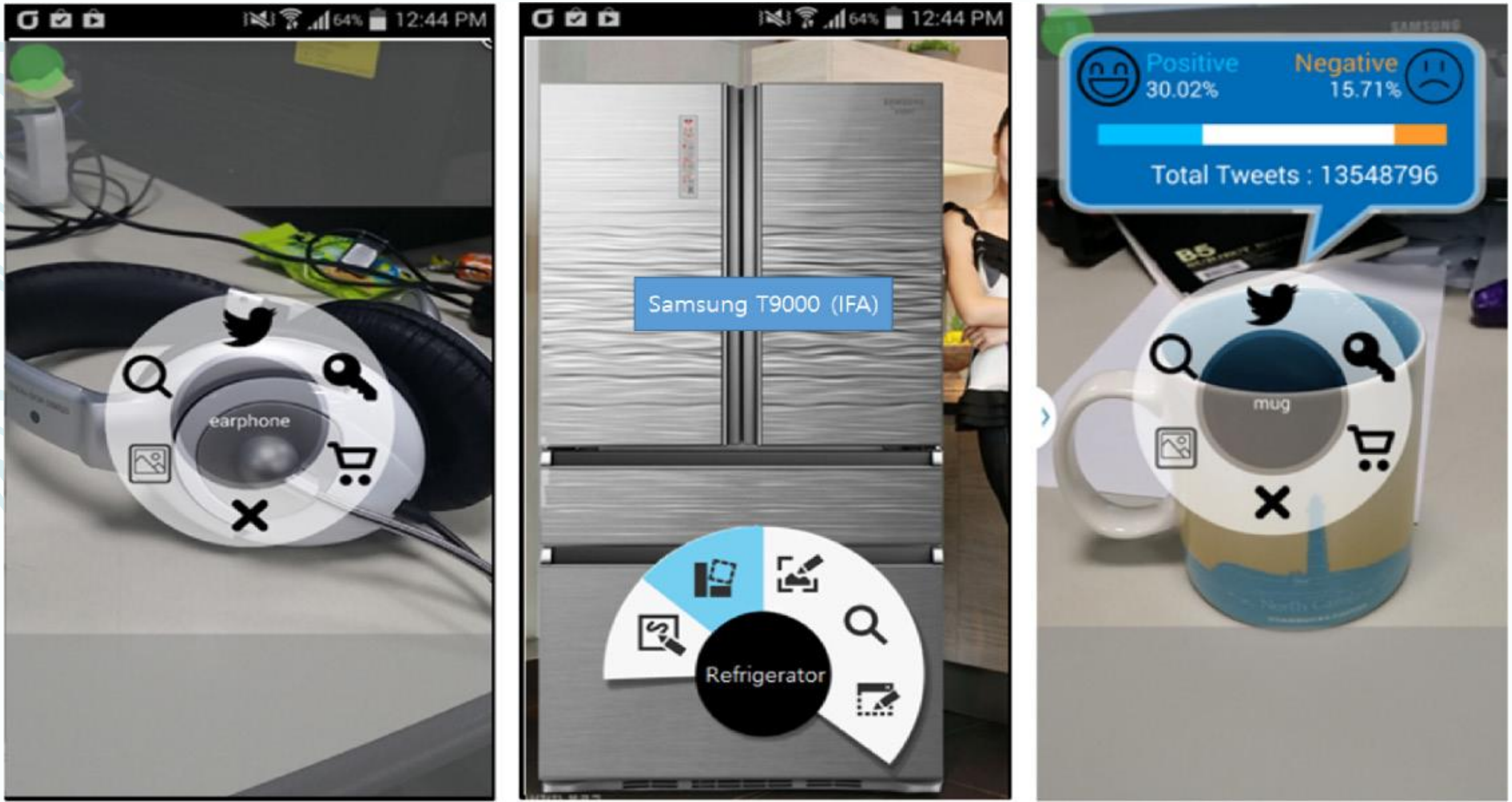
- The above Figure, demonstrated the implementation of the device/cloud collaboration framework by the semantic QA cache.
- Specifically, we have devised a custom ASR (Automatic Speech Recognition) engine for the mobile device and incorporated the cloud system for Samsung S Voice in the collaboration framework.
- The cloud system for S Voice consists of a Natural Language Understanding (NLU) module for query understanding, a DM (Dialog Manager) module for query answering, and a powerful ASR engine.

- Note that we have adapted the framework to compute the probability of the on-device semantic QA cache to answer a given query correctly.
- If the probability is high enough, the Device/Cloud Selection module will take the risk of looking up the semantic QA cache for an answer.
- If the cache does not return the right answer and forces the user to ask the cloud again, then our framework will adjust the probability accordingly.

IMAGE AND SPEECH RECOGNITION

- Automatically recognizing images and speech can greatly enhance a user's experience in using applications.
- For example, with automatic image recognition, photos taken by a user can be automatically tagged with metadata and catalogued more easily.
- An application called Watch&Go, which lets users obtain detailed information about a product upon taking a photograph.
- The following Figure shows the snapshot of Watch&Go that guides users to properly focus on some electronics products, and automatically retrieve information such as type, vendor, model name, and the result of social sentiment about the product.

An Example of Automatically Tagging Recognized Images and Displaying Additional Information Such as Social Sentiment (eg, Positive or Negative Reviews)



- Practicality of these recognition applications has greatly improved, thanks to the recent advancement of Deep Learning (DL).
- The DL follows the approach of learning the correlation between the parameters across multiple layers of perceptron.
- However, DL model training methods usually suffer a slow learning curve compared to the other conventional machine-learning methods.
- Although it is generally believed that the larger DL model improves the recognition accuracy through a set of well-refined training data, it has been challenging to acquire adequate parameters when we train multiple layers at the same time.
- The recent appearance of the Restricted Boltzmann Machine (RBM) method, which enables layer-wise and unsupervised training, can relax the aforementioned limitations to some degree.
- However, the overall computational overhead is still formidable, even for the cloud with abundant compute resources.