Lucida

November 22, 2016

Directory

- Overview
- Introduction:

  Infrastructure for Intelligent Web Services
- Architecture
- Features:
  - Consistency
  - Flexibility
  - Scalability
- Demo

Overview

- Lucida is a speech and vision based intelligent personal assistant (IPA) developed in Clarity Lab at the University of Michigan. It is a state-of-the-art infrastructures to study emerging intelligent web services in large scale systems.
- Web: http://lucida.ai
- Github: https://github.com/claritylab/lucida
- Google Group: https://groups.google.com/forum/#!forum/lucida-users

Introduction: Infrastructure for Intelligent Web Services

- As Intelligent Personal Assistants such as Apple Siri, Google Now, Microsoft Cortana, and Amazon Echo continue to gain traction, web-service companies are now providing image, speech, and natural language processing web services as the core applications in their datacenters. These emerging applications require machine learning and are known to be significantly more compute intensive than traditional cloud based web services, giving rise to a number of questions surrounding the designs of server and datacenter architectures for handling this volume of computation.

Introduction: Infrastructure for Intelligent Web Services

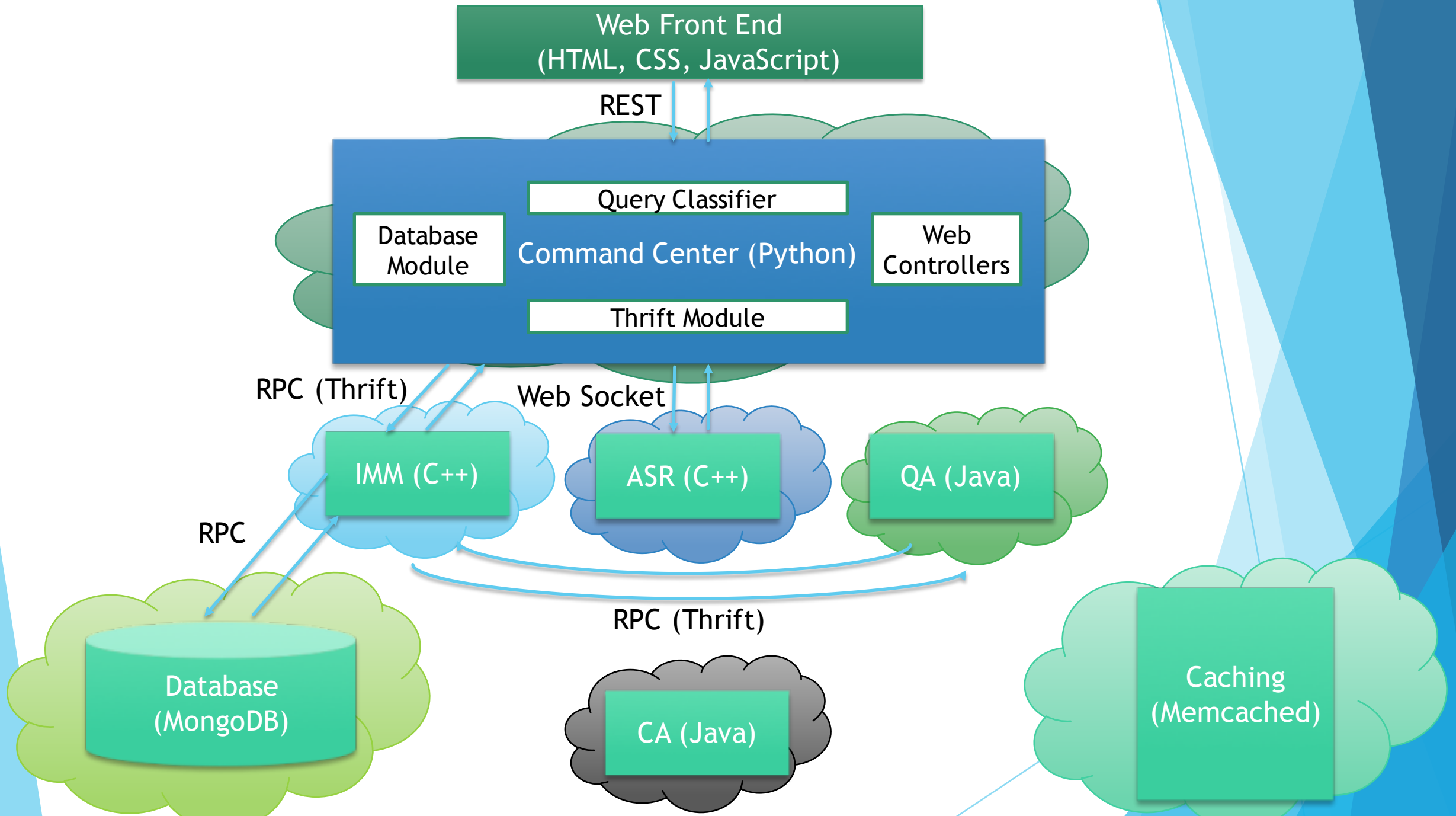
- Lucida is the next generation of Sirius, the first open source intelligent personal assistant that was developed by Clarity Lab at the University of Michigan.

Introduction: Infrastructure for Intelligent Web Services

- Research questions arise as complex systems with multiple AI services are built.
- On the system side, how to combine individual components into an eco-system with easy scalability and maximum flexibility? How to dynamically scale service instances up and down based on the current load?
- On the AI side, how does the accuracy of each individual component affect the end-to-end accuracy which decide? How to choose each AI component so that the end-t-end accuracy is maximized?

Architecture: List View

- Web Front End
- Command Center (CMD)
- Core AI services:
  - Image Matching (IMM)
  - Automatic Speech Recognition (ASR)
  - Question Answering (QA)
  - Calendar Events Retrieval (CA)
  - Digit Recognition (DIG)
  - Face Recognition (FACE)
  - Image Classification (IMC)
- Database
- Caching


Web Front End
(HTML, CSS, JavaScript)
REST
Query Classifier
Database Module
Command Center (Python)
Web Controllers
Thrift Module
RPC (Thrift)
Web Socket
IMM (C++)
ASR (C++)
QA (Java)
RPC
RPC (Thrift)
Database (MongoDB)
CA (Java)
Caching (Memcached)

Architecture:
Command Center (CMD)

- The service manager that connects the back end to the front end.
- Written in Python, involves the following major modules:
  - Web Controller: Servers web pages;
  - Database: Stores and retrieves from the database component;
  - Caching: Stores and retrieves from the caching layer;
  - Query Classification: Classifies input text query into a set of services needed;
  - Thrift: Sends query to and receives response from back-end services through Thrift, an RPC framework supporting multiple languages.

Features

- Consistency
- Flexibility
- Scalability

Feature: Consistency

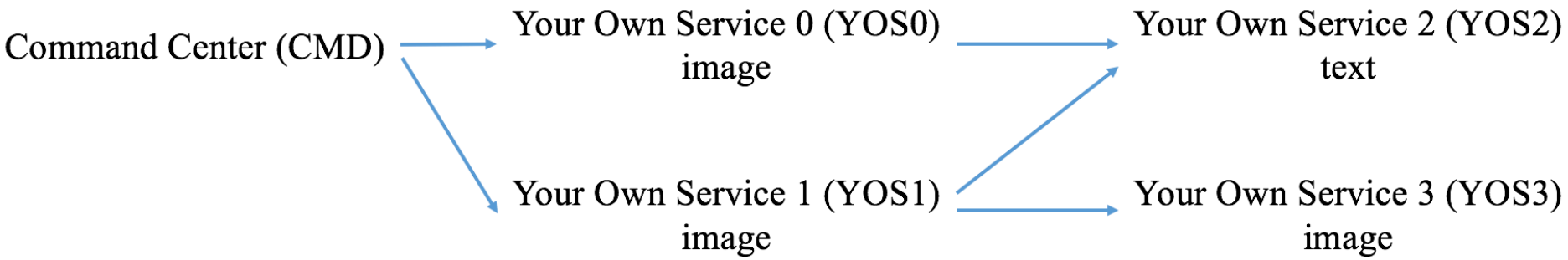
- Backend services communicate with each other through Thrift.
- Standardized interface to be implemented by backend services:

```
include "lucidatypes.thrift"
service LucidaService {
    void create(1:string LUCID, 2:lucidatypes.QuerySpec spec);
    void learn(1:string LUCID, 2:lucidatypes.QuerySpec knowledge);
    string infer(1:string LUCID, 2:lucidatypes.QuerySpec query);
}
```

- See details in the top-level readme.

Feature: Flexibility

- Arbitrarily complex service combinations can be specified to CMD.
- CMD sends a service graph to backend services:



- Services are responsible for further routing queries to other services.
- All the complexities are wrapped in the standardized interface.
- See details in the top-level readme.

Feature: Scalability

- Rely on Docker and Kubernetes for easy and scalable deployment.
- Specify the number of instances when services are started:

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  labels:
    name: asrworker
  name: asrworker-controller
spec:
  replicas: 2
```

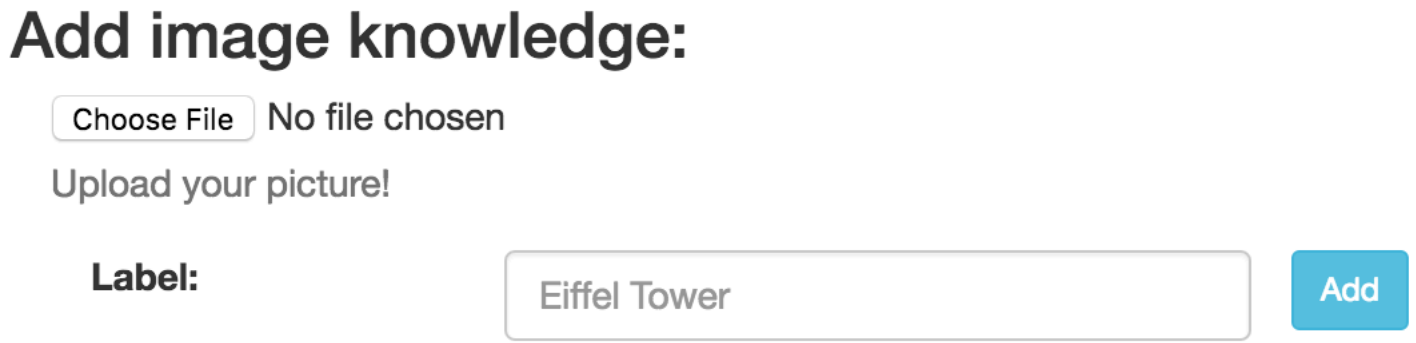
- See details in tools/deploy.
- In future, can support automatic scaling based on current load from CMD.

Demo

- Set up the Lucida eco-system following “**Lucida Local Development**” or “**Lucida Docker Deployment**” of the top-level readme.
- In the next slides, we are going to demo:
  - Image Matching
  - Question Answering
  - Image Matching + Question Answering
  - Calendar
  - Digit Recognition
  - Face Recognition
  - Image Classification

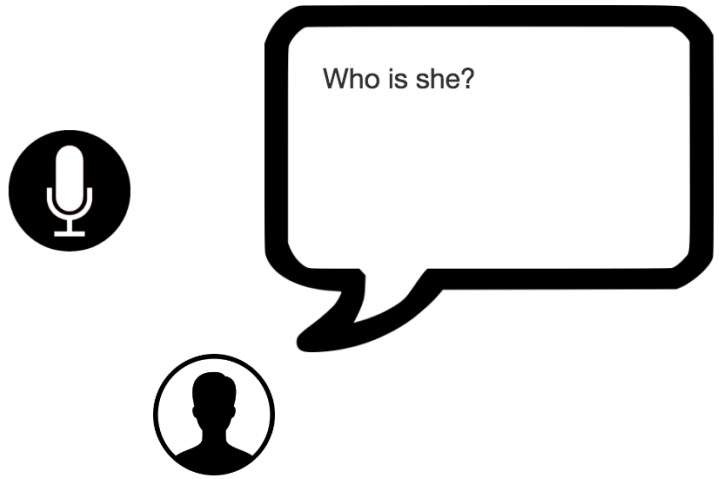
Demo
Image Matching

- 1. Add at least one image under the “Learn” tab.



Demo
Image Matching

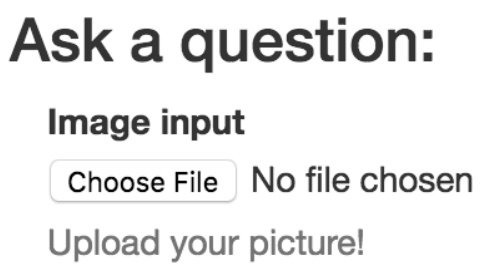
- 2. Navigate to the “Infer” tab. Ask one of the following questions:
- Who is he?
- Who is she?
- Who’s in the image?
- Match this image.
- Which photo in my collection is similar to this?



- In fact, this step is optional. You can directly go to step 3.

Demo
Image Matching

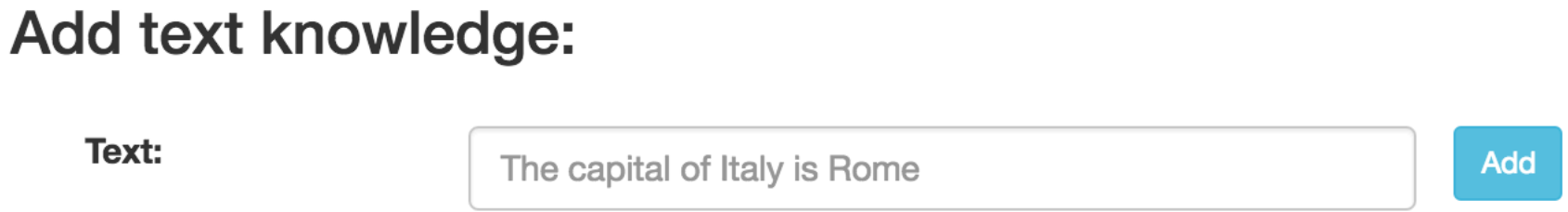
- 3. Upload one image that you want to match.



- 4. Click the “Ask” button.

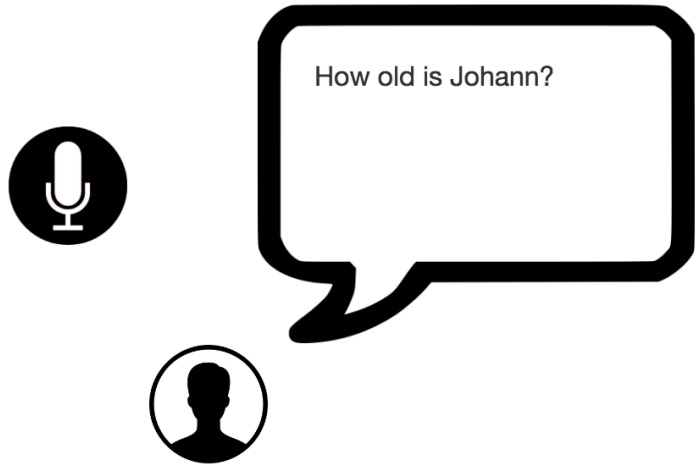
Demo
Question Answering

- 1. Add at least one piece of text under the “Learn” tab.
- For example, “Johann is 25 years old.”



Demo
Question Answering

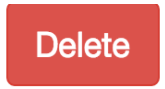
- 2. Navigate to the “Infer” tab. Ask one question related to the knowledge you added in step 1.
- For example, “How old is Johann?”
- 3. Click the “Ask” button.



Demo
Question Answering

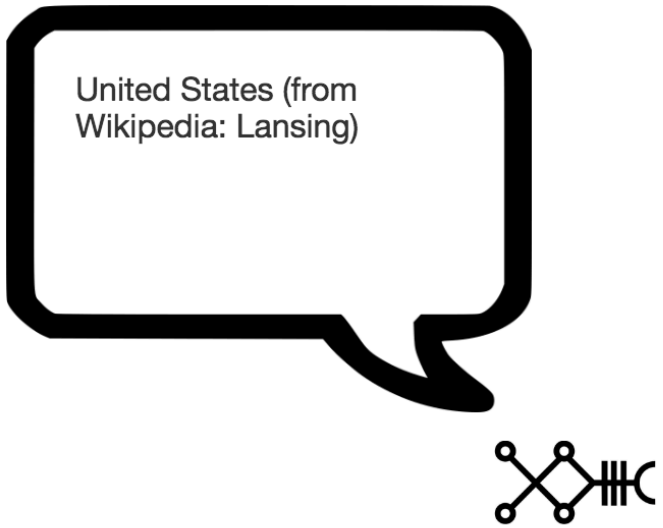
- 4. Go back to the “Learn” tab. Add a Wikipedia page URL:
- For example, https://en.wikipedia.org/wiki/University_of_Michigan

  - https://en.wikipedia.org/wiki/University_of_Michigan



Demo
Question Answering

- 5. Go back to the “Infer” tab. Ask a question related to the knowledge you added in step 4:
- For example, “Where is the University of Michigan located?”
- 6. Click the “Ask” button.

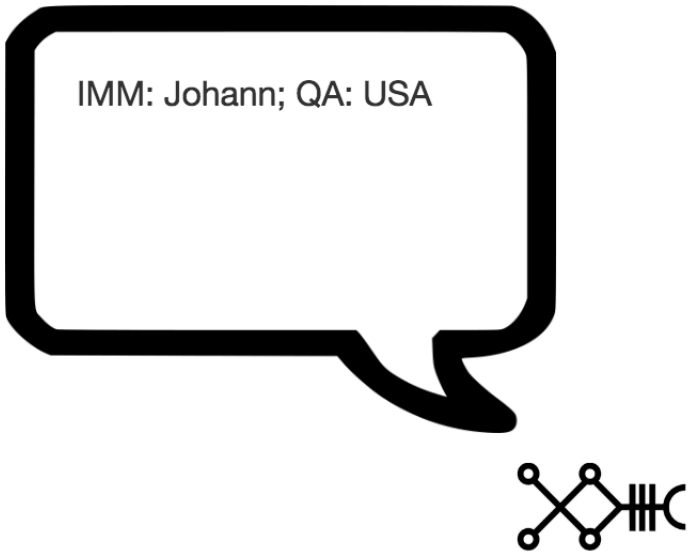


Demo
Question Answering

- 7. Ask one of the following questions:
- Who is the last emperor of China?
- Who discovered the South Pole?
- Who invented the telephone?
- Who is the Greek mathematician that developed geometry?
- When was the Munich agreement made?
- When did Mike Tyson win the title?
- Where is the River Nile located?
- 8. Click the “Ask” button.

Demo
Image Matching + Question Answering

- 2. Navigate to the “Infer” tab. Ask the following question and upload the image with “Johann” in it.
- `Where was he born?`
- 3. Click the “Ask” button.

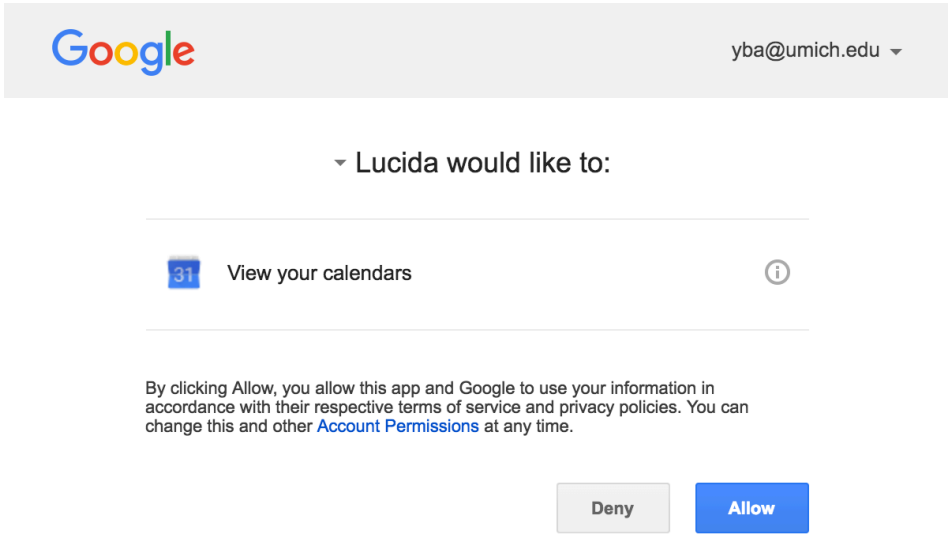


Demo
Calendar

- 0. We assume you have some events on your Google Calendar.
- 1. Navigate to the “Infer” tab and ask one of the following questions:
- `What am I going to do tomorrow?`
- `What is my plan next Monday?`
- `What is on my schedule next Tuesday afternoon?`
- `What did I do last weekend?`
- `What was on my Google Calendar last year?`
- `What was on my schedule last Christmas?`
- 2. Click the “Ask” button.

Demo
Calendar

- 3. Allow the pop-up window and click “Allow”:



Demo
Digit Recognition

- 1. Navigate to the “Infer” tab and ask one of the following questions:
- `What digit is in the image?`
- `Lucida, what is this number?`
- 2. Upload one of the following JPEG images:

- 3. Click the “Ask” button.

Demo
Face Recognition

- 1. Navigate to the “Infer” tab and ask one of the following questions:
- `Who is this famous star?`
- `Do you know this celebrity?`
- 2. Upload an image of JPEG format with a celebrity in it.
- 3. Click the “Ask” button.

Demo
Image Classification

- 1. Navigate to the “Infer” tab and ask one of the following questions:
- `What is this thing?`
- `What is the animal?`
- 2. Upload one of the following JPEG images:



3. Click the “Ask” button.

Thank you!

- Please leave your feedback in our Github repository or Google group!