Tiki Systems White Paper

Tiki Systems: An end-to-end implementation of Question Answering using Video and Voice

Thematic Areas: ☐ Pythia ☐ IoT ☐ Meteor ☐ UIX

☐ VQA ☐ Deep Learning ☐ Transfer Learning ☐ Cloud Servers

☐ Google Speech-to-Text

Principal Authors:

Name: Brent Biseda

Name: Vinicio DeSola

Name: Pri Nonis

Name: Kevin Stone

Institution: School of Information - University of California, Berkeley. W251

Contents

Abstract:

The advancements in Deep Learning have changed many areas of research and applications, going from Image Recognition to Natural Language Processing. One of the newest frameworks in *Pythia*, developed by Facebook AI Research, that allows for VQ&A (Visual Question and Answer). In this project, we developed a end-to-end implementation of this framework, with a friendly UIX for the user, and with integration with Google Speech-to-Text for better integration, all wrapped up in a website calles `www.tiki.systems`. This prototype sets the ground for many other useful applications of VQA with Speech-to-Text integration.

1 Introduction:

Curiosity is the main force of creation. And one of the main drivers of curiosity is asking questions. Not long ago, students, professors, etc. had to go to libraries or use encyclopedias to find the answer to any question they had. All this changed with the introduction of the Internet: a pool of almost infinite knowledge that has become easier and easier to query. Nowadays, if you have a question, you just *Google* it.

Question and answering have become an important and fascinating area of research in the last few years: *Google* evolved from a web searcher to a well-oiled machine of answering almost all written queries that you might have. One of the main sources of this evolution, is the introduction of algorithmic ways for a computer to answer these questions without human input. Deep learning in particular has become the main tool used for this area of research.

Deep learning started gaining the attention of the Image Recognition field, where it was discovered that Convolutional Neural Networks (CNN) were able to apply filters that initially were capable of recognizing edges, and then, when stacked, more and more complex shapes such as faces, animals, etc. The next area that was disrupted by Deep Learning was Natural Language Processing (NLP): using many neural architectures, we are now capable of predicting language patterns, accurately translate between languages, and even start answering questions via language predictions.

In this white paper, we will describe our implementation of several tools that will help us

answer questions about an image, either by typing the question or by asking it via a microphone: closing the bridge between image recognition and NLP. We use *Pythia*, a new Facebook framework for VQ&A that allows the user to ask written questions about an image. We decided to implement an app that allows the user to either use his/her camera and microphone to ask Tiki, our app, any question about the Tiki's video feed. The prediction is done on a cloud server (V100 for faster inference), and the questions and answers are stored for future reference.

2 Motivation:

Many of the Deep Learning frameworks out there have interesting tutorials with toy examples: stock images or images from the web. However, to bring theses tools to a production level of functionality, we need more than a toy example: enter `www.Tiki.systems`.

We connected all elements of a deep learning framework into a production-level setting with an end-to-end implementation: get the image and the question from the user, pass it to the cloud server where the inference is conducted and bring back the answer to the user. We are using a state-of-the-art framework called *Pythia*, where documentation is scarce, involving a data engineering challenge in itself. Finally, connecting the user to the framework in a easy and intuitive way is not straightforward, needing UX design and expertise to deliver a complete prototype.

3 Architecture Design:

In this section we describe the end-to-end architecture that we are using. We are leveraging a pre-trained framework of prediction by using *Pythia*, but we also need other tools to bring the prediction to the final user as seen in Figure 1.

Our first main goal, is to be accessible: thus Tiki works directly in the Browser (any browser) and does not require the user to install any additional software. To achieve this, we require the use of the following front-end applications:

1. **Meteor**: Full-stack JavaScript App that assembles all the pieces needed to build either web-sites or mobile apps. It automatically manages the data flow between cloud and client applications, client UI and rendering, no matter the framework used. It supports Angular, React, and its own

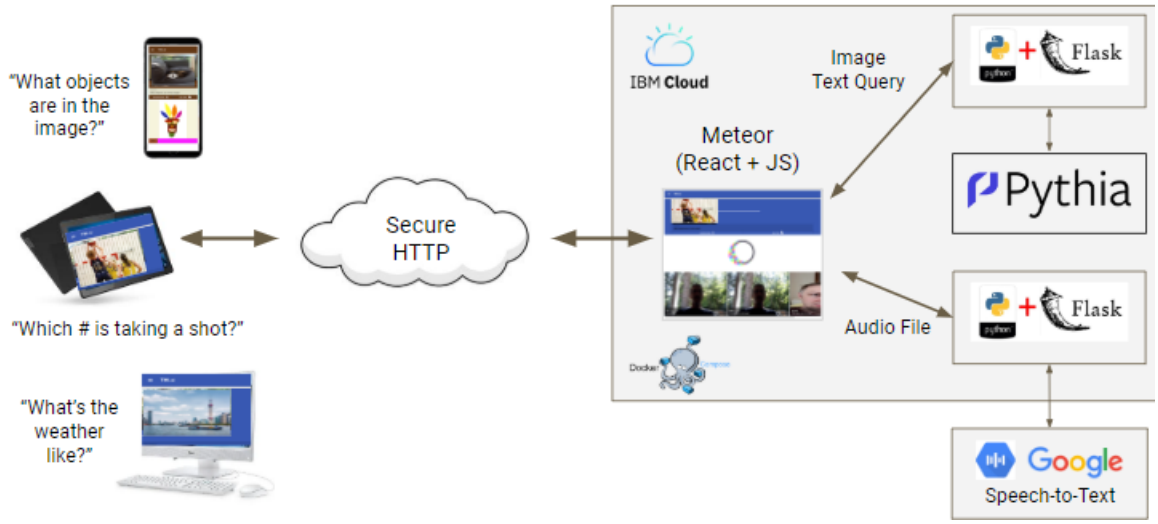

Figure 1: Tiki Systems Architecture

internal library called Blaze (4).

2. **React**: React is a declarative, efficient, and flexible JavaScript library for building user interfaces, using a modular framework called components (6).

3. **Material Design**: Leading open-source design system developed by Google, and allows designers to create better UI for the user in a practical, usable, and pretty way. (3)

The back-end applications used for Tiki are described below with the most important one being *Pythia*.

1. **Flask**: Lightweight web application framework, designed for quick and easy learning curve, but with capabilities to scale up to complex infrastructures. Based on Python, it offers support to the developer, but doesn't enforce any dependency or layout (5)

2. **docker-compose**: Tool develop by Docker to run several containers at the same time. The compose file is a YAML file with all the instructions and details of each container that the application will use, and then, with a single command we can open all the respective containers (2).

2. **Pythia**: Developed by Facebook AI Research, is designed for answering questions related to visual data and automatically generating image captions. It has several application modules: Question Encoding, Image Feature Extraction, fusion of the last two with an attention head, and Classification over the space of the answers. It is built on an open-source Framework.

Pythia's critical idea is a bottom-up and top-down attention model. It uses an object detector (a

Faster RCNN pre-trained on the Visual Genome dataset) to extract images features with bottom-up attention, as seen in Figure 2. For combining Image with text, it does element-wise multiplication to combine the features from both spaces. The Questions are represented by using GloVe, and then pass the word embedding to a modified LSTM network (7). In terms of performance, it achieved a 70.34% accuracy on VQA 2.0.

3. **Google Speech-Text**: Used to process real-time streaming or with prerecorded audio. It's hosted on the Google Cloud in an easy to use API. It is based on an RNN-T transducer, which reduces network latency, and speeds up the recognition of the language: it supports over 120 spoken languages. The RNN-T architecture model can be seen in Figure 3 (1).

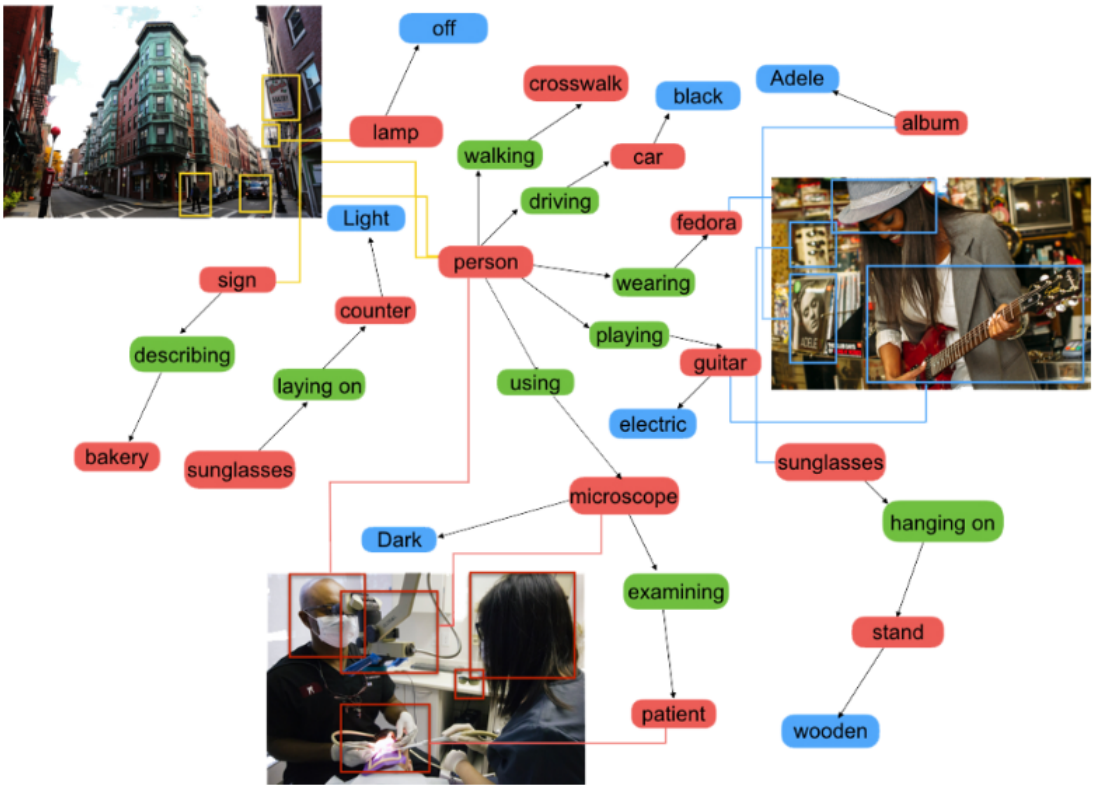

Figure 2: Attention Model of Pythia

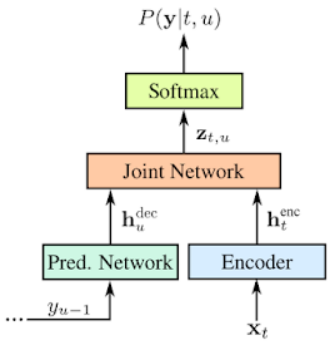

Figure 3: Final feedback loop of the RNN-T Architecture

4 Tutorial of Tiki

In this section we will explain how to use `www.tiki.systems` and how each component works on the back-end.

1. **Go to the url, either on your PC or your mobile phone**: One of the many advantages of using our previously described architecture, is that the user can log into the app from any browser, without any installation process. In Figure 4 we can see the PC's landing page.

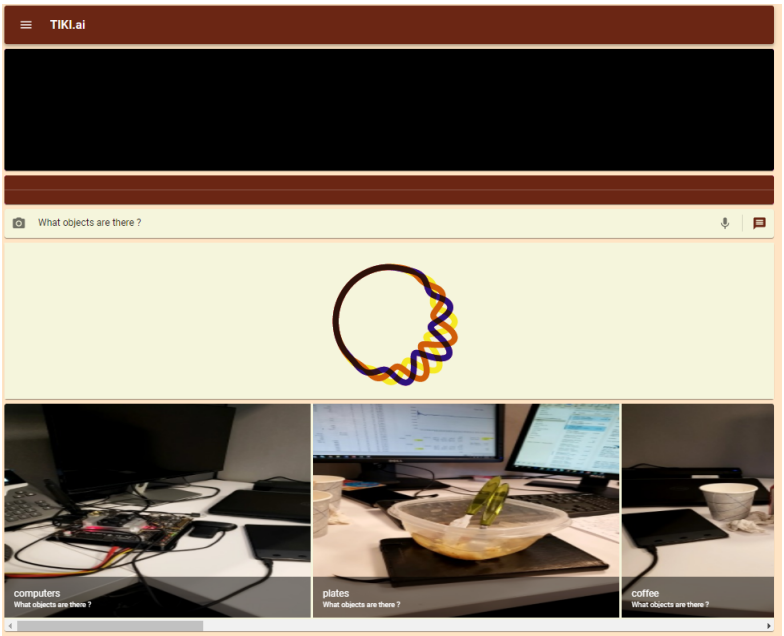

Figure 4: PC's Landing Page

2. **Video Feed**: Once the user lands on the url, the browser will immediately ask for permission to use the camera of the device, which will be projected on top. Once the video starts, the user can either write the question into the text bar (Default question: *What Objects are there?*) or press the microphone button to ask the question. When the user presses the microphone, we call Google's Speech-to-Text API, which will infer the question from the voice of the user and write it into the text bar. This is an independent inference from Pythia, and is done using Google Cloud services.

3. **Ask TIKI**: Once the user types the question (either by text or by using the microphone), she/he needs to *Ask TIKI* by pressing the message button. This is where the inference process begins. In Figure 5 we can see how the internal Pythia Inference Process works.

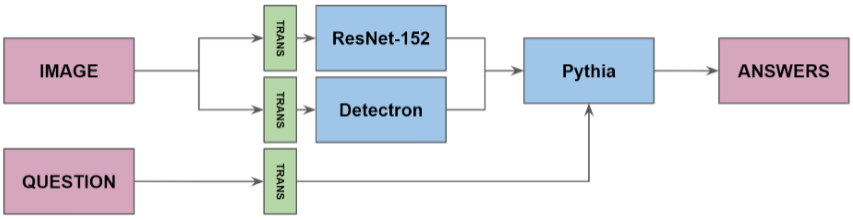

Figure 5: Pythia Inference Process

The inference is run on a V100 in the IBM Cloud servers, for faster inference and lower latency for the user. In Table 1, we present the average time of initialization and inference on the V100:

Process	Time
Initialization	16.125 seconds
Inference	1.5 seconds

Table 1: Time Performances

4. **Output**: Once inference is finished, the user will get a table of probabilities with the most likely answers to his/her question, as seen in Figure 6. Then, images are labeled with the top answer for the question and stored on the bottom of the landing page. We save and display 10 images on the landing page.

Rank	Answer	Probability
0	office	94.55 %
1	classroom	1.71 %
2	kitchen	1.02 %
3	hospital	0.66 %
4	bathroom	0.39 %

Figure 6: Table of results

5 Future Work

In this section we will describe future work that can be done with our prototype:

1. **Near-term**:

- UI Updates: As with any website, we can continue to make improvements on how the UI looks, and how to enhance the user's experience with our app.

- Medium Article: Because Pythia is such a new framework, documentation is scarce, in fact, one of the many challenges of this project was to learn how to set up Pythia in our servers and do inference with it. Thus, we will write a Medium Article with sample codes to help future users of it.

2. **Long-Term**:

- Integration with smart home devices API: Alexa / Google Home / Nest Cameras.

- Specific Training: We are using the out-the-box pre-trained models from Pythia, but the framework is flexible enough to allow for training to specific tasks: Home Surveillance, Healthcare, etc. Also, we can add different word embeddings, like domain-specific BERT models, for better understanding of the context of the questions.

6 Conclusion

We were able to connect Pythia to a website UI, with integration to Google's Speech-to-Text model, obtaining interesting results in real-time VQA problems. With this prototype, we have a proof of concept of what can be done in this space thanks to this new open-source frameworks.

References

[1] Google AI. An all-neural on-device speech recognizer, Mar 2019. URL https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html.

[2] Docker. Overview of docker compose, Aug 2019. URL https://docs.docker.com/compose/.

[3] Google. Getting started, 2018. URL https://material.io/collections/getting-started/#01.

[4] Meteor. Meteor, 2018. URL https://www.meteor.com/meteor-faq.

[5] Pallets. Flask, 2010. URL https://palletsprojects.com/p/flask/.

[6] React. Tutorial: Intro to react, 2019. URL https://reactjs.org/tutorial/tutorial.html.

[7] Amanpreet Singh, Vivek Natarajan, Yu Jiang, Xinlei Chen, Meet Shah, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. Pythia-a platform for vision & language research. In *SysML Workshop, NeurIPS*, volume 2018, 2018.