Content association service for operating IoT devices in a Smart Home Environment

Ankit Jain
Samsung R&D Institute Bangalore, India
jain.ankit @samsung.com

Siba Prasad Samal
Samsung R&D Institute Bangalore, India
siba.samal @samsung.com

Ravi Nanjunappa
Samsung R&D Institute Bangalore, India
nravi.n @samsung.com

***Abstract* - Context and content that a user goes through often drives the operation of IoT devices taking minimal leverage on automation processes resulting in poor user experience and underutilization of the IoT ecosystem's potential. IoT devices in smart environments are becoming more intelligent through AI and machine learning enabled architectures. They have begun to progressively interweave with a user's life, be it decision making or letting users drive and complete their regular activities like cooking through these IoT devices and cloud service enablers. With the exponential growth and users adoptions, there is a dire need of having content providers integrate their content with actionable components on the user's IoT devices. To enable this, a system is required to give content providers a mechanism to provide IoT enabled content which can leverage a user's IoT ecosystem. We propose a 2 tier architecture wherein a web service correlates relevant content with a possible IoT device interaction through semantic analysis driven prediction engine and introduce new xml tags to embed these probable IoT Interactions in the web page without making the content providers modify their content. The 2nd component of the architecture consists of an on-device service that could relate this information with the user's IoT ecosystem involving further conversion of the IoT interaction to fit a user's IoT devices and commands. This approach ensures privacy awareness in the user's IoT ecosystem while still giving content providers a system to make their content IoT enabled with an unknown IoT ecosystem.**

Keywords-Smart Home, IoT ,Web Service, Privacy Awareness, Hybrid Service, Machine Learning, Device

I. INTRODUCTION

IoT devices are becoming exponentially popular across smart home, industry, and vehicle to vehicle communication environment. With the recent 5G network implementation across the world, the use cases of the IOT, artificial intelligence are just going to grow substantially. With the increasing devices inside Smart home there will be a demand on seamless usage of these devices along with various other applications and contexts.

The IoT system for Smart Home mainly consists of four components [Fig 1]. The first component resides on the IoT device itself which connects either directly to cloud or connects to the home gateways. The second optional component resides on the home gateways which acts as a mediator between end IoT devices and Cloud. It forwards the event information of the IoT devices to cloud and controls the devices if request is received from the cloud. This component might not be needed for the IoT devices which are capable of connecting directly to the cloud. The third component constitutes of Cloud services which enables the devices which can be controlled from anywhere and anytime. These services includes logging the device events, executing automations and advanced services such as energy saving, elderly care, routine generations etc. The fourth components resides in the form of IoT applications in user devices such as smartphone, TV and wearables from where user can control her IoT devices from anywhere.

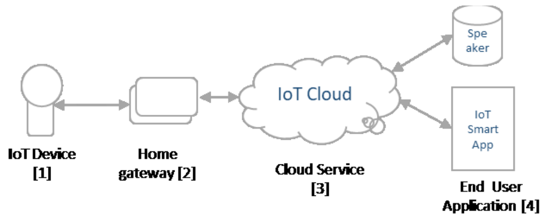
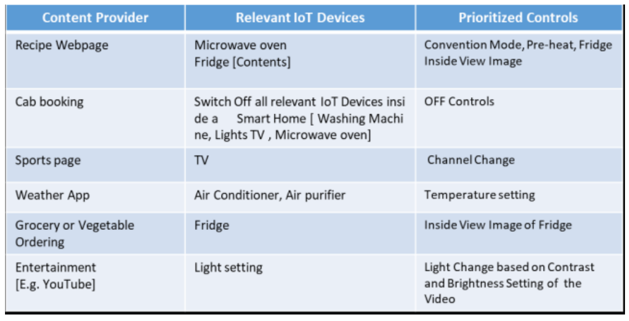


Figure 1. High level IoT Components for Smart Home

Traditionally, with the evolution of www and mobile applications, many content providers are providing contents to the users and some of these contents are related to the user's IoT devices. Table 1 shows some of the web pages or applications related to travel, food recipes, weather, grocery ordering, exercise, entertainment that are directly related to popular IoT devices inside a Smart Home environment. To provide a seamless experience in operating relevant IoT devices from the content, this paper proposes a novel service based architecture and content association algorithm.

TABLE I. EXAMPLE OF THE CONTENT RELEVANCY WITH IOT DEVICES AND CONTROL



Content Provider	Relevant IoT Devices	Prioritized Controls
Recipe Webpage	Microwave oven Fridge [Contents]	Convention Mode, Pre-heat, Fridge Inside View Image
Cab booking	Switch Off all relevant IoT Devices inside a Smart Home [Washing Machine, Lights TV , Microwave oven]	OFF Controls
Sports page	TV	Channel Change
Weather App	Air Conditioner, Air purifier	Temperature setting
Grocery or Vegetable Ordering	Fridge	Inside View Image of Fridge
Entertainment [E.g. YouTube]	Light setting	Light Change based on Contrast and Brightness Setting of the Video

The rest of the paper is divided as follows. Section 2 explains related work that has been done, section 3 proposes a new architecture for seamless usage of IoT devices from various application contexts, section 4 describes the core algorithms used for identification of the IoT devices from related contents, section 5 describes the simulation and evaluation and section 6 concludes this paper. References have been added at the end of the paper.

II. RELATED WORK

In the recent times, there is an exponentially growth of new IoT devices inside smart home and outside. Current research focuses on how to handle the scale of the devices and optimized IoT middleware to handle these devices intelligently.

Semantic Web Techniques are currently used for IoT middleware. [1] Suggests an architecture which leverages semantic web techniques for Global Sensor Network by the use of an interoperable and configurable middleware. [2] Paper proposes prolog reasoning based task planning mechanism for Smart Home based on Multi-Agent System ontology for smart home and semantic annotation of resource information. [3] EUPont is a Semantic Web ontology users enable to meet their needs with fewer and higher level rules which can be adapted to different contextual situation.

Combining web services along with cloud computing with the Smart home environment is discussed in [8]. Various features including of context-awareness, energy efficiency, natural interaction, and user activity recognition are proposed using web services. IoT devices operation behavior was predicted using heterogeneous network and Cloud computing [9]. On similar lines controlling and monitoring home equipment's and devices remotely is proposed via Web Services [11].

On-device screen intelligence engine is proposed [6] where text data available on a user's device along with the application type context is used to classify and predict relevant IoT devices and control. This work uses a ML based approach for understanding user intent from the contents of the screen and recommends suitable devices and actions. This approach needs extraction of data from images. [4] and [5] explains details about the extraction of the useful information from the images but since in this paper we were mostly dealing with web contents we would easily get the textual data required for processing from the web content itself.

In this paper we focus on bringing a correlation between Smart home devices and the content consumed by the various user devices specifically Smartphones, televisions or other display enabled devices. The proposed architecture is flexible and can be easily integrated with current web contents.

III. OVERALL ARCHITECTURE

We propose a 2 tier architecture, where one component of the architecture (termed as "IoT device parser" service) runs on the cloud and finds the probabilities of correlations of content from content providers to IoT devices and device commands. The service then is responsible for injecting appropriate custom IoT HTML tags in the page. The second component of the architecture (termed as "IoT Device mapper") runs on the user's device where the user consumes content and this component links the IoT devices in the user's ecosystem to the suggested / predicted IoT devices on the content provider's page (injected through custom HTML tags).

A. IoT device parser service (Cloud / Content Provider's Side)

This service (can be a micro service) will run on the cloud. Figure 2 shows the modules that make up this service where content recognition engine is responsible for associating IoT devices with the content. The service has two major steps, firstly it will analyze and correlate the content given to this service with relevant IoT devices and commands as proposed in the algorithm in following section. Second step involves formation of relevant IoT custom HTML Tags by one of the service modules and injection of these tags in the given HTML page or returning the predicted IoT devices and commands as response.

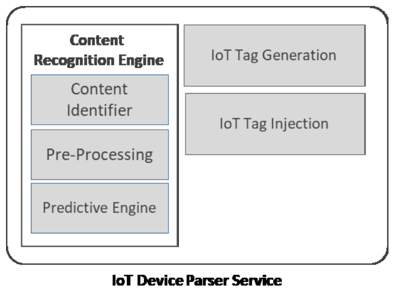


Figure 2. Modules comprising of IoT device parser service

There are 2 ways that the content providers currently use to render content in user web browsers:

1) Server side rendering of HTML pages and optionally caching these pages.

2) Client side rendering of HTML pages using server api calls to populate and change content.

1) Service Side rendering of HTML pages optionally caching these pages.

Figure 3 shows the working of IoT device parser service under the given situation. In this method the content providers forms the complete html page on the cloud / server with the content and then this formed html page is given back as response to a client browser. Our proposed architecture involves the content providers to pass this formed html page to our service along with the top element's unique selectors whose child's have the actual content. An HTML elements unique selectors helps the system to get the exact position of the element in the HTML page and retrieve the content inside that element. This helps the service to locate the IoT related content as otherwise if the service had to itself find the relevant content, then the computation time of the service would increase.

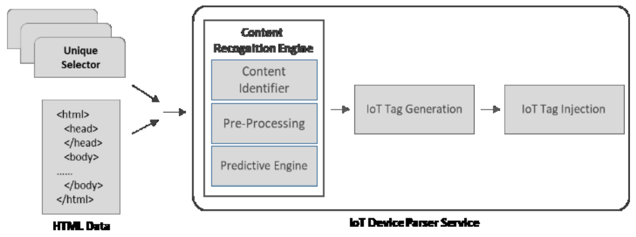


Figure 3. IoT device parser service running in Server side rendering of html scenario

The content identifier module is responsible for stripping element tags from the content and combining content together to pass to processing. It also removes certain tags with inside content, like tags which have “img”, “audio”, “video”, “svg” element inside it, elements like “blockquote” etc. The content is combined by following inline and block elements description of HTML tags, where content inside block elements are send as different units for processing. Content inside Inline elements are merged with other surrounding elements. The output from content identifier module is textual without html tags since this text has to be processed.This text then is passed through the pre-processing module which processes the data according to the steps defined under the text preprocessing section.

After pre-processing step the content is passed to the predictive engine with actually performs the processing to relate IoT devices and commands on those IoT devices with the given content. These IoT devices and Commands are then passed along to “IoT Tag Generation” module which forms relevant tag hierarchy of the predicted IoT devices and commands. These tags are then inserted in the HTML page received from the content provider by “IoT tag Injection” module which associates these tags with parent level HTML tags as these tags wraps the content to which the predicted IoT devices and commands belong to. To create this association “IoT tag injector” gets unique selectors to the parent level HTML tags and adds them as one of the attributes to custom IoT elements. Then the entire hierarchy is appended either at last of the parent elements or at end of the html page.

2) Client side rendering of HTML pages using server api calls to populate and change content.

Fig 4 shows the flow of IoT device parser service in Client side rendering situation. In this method the content providers only give minimal html document required to render the layout in browser side. The actual displayable content is fetched by JavaScript through XHR requests to the server where the server response contains content data and this data is used to create HTML elements on the fly on the client browser based.

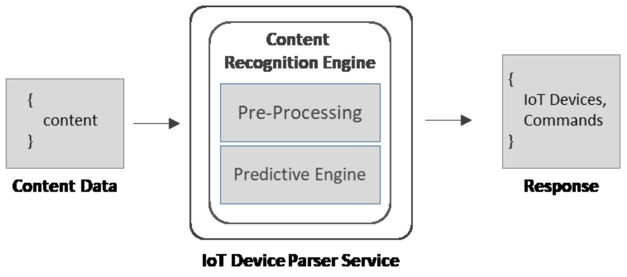

{ content }
Content Data
Content Recognition Engine
Pre-Processing
Predictive Engine
IoT Device Parser Service
{ IoT Devices, Commands }
Response

Figure 4 IoT device parser service running in Client side rendering of html scenario

To get content associated IoT devices and commands the client side sends the relevant content from server directly to the “IoT device parser service” and the service in this case passes the input to the “pre-processing” module then to “predictive engine” and then the predicted IoT devices and commands are directly given to the browser which are then associated with the content along with appropriate tag generation. The process of making a request to the web service and associating content with the response of the service is done by “IoT device mapper” component. Such an approach ensures that the content provider have to do minimalistic changes to their websites to leverage a user’s IoT ecosystem.

B. *IoT Device Mapper (Client side)*

This service is run on a client’s smartphone device or other similar IoT devices where browser support is present. The service is run as a browser plugin and interacts with an IoT ecosystem provider like an application or system service. The data exchange to share knowledge of a user’s IoT ecosystem can happen over web socket, http or https.

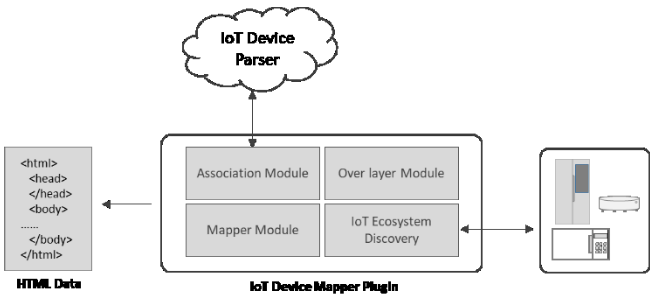


Figure 5. IoT device mapper plugin modules and interactions

The plugin is responsible for making client-side integrations easier as the plugin communicates with the “IoT device parser” service and associates content with IoT devices and commands. Once the plugin gets IoT devices and commands either by interacting with the web service or by parsing custom tags present in html, similar IoT devices having similar commands are mapped from User's IoT ecosystem to the content as the plugin is aware about a user's IoT ecosystem. The plugin is further responsible for deciding on an appropriate interface to make it easier for users to access their IoT devices and commands on those devices directly from the content or from the web page. These tasks are divided in 4 different modules in the plugin:

1) *Association module*
2) *Overlay module*
3) *Mapper module*
4) *IoT ecosystem Discovery module.*

1. Association Module

This module is responsible for interacting with the "IoT device parser" web service. The client calls API present in this module to let the plugin know the existence of new map able content. This module takes the content and sends it to the web service. After getting relevant IoT devices and commands from the web service, the plugin associates it with the displayed content when the content is displayed in the viewport.

2. Overlay Module

This module is responsible for giving accessible interfaces to a user for interaction with IoT devices and commands. It is also responsible for managing that interface and interacting with IoT ecosystem discovery module for user action propagation. The module gives 2 types of interfaces depending on the placement of custom IoT tags. In one of the interfaces the controls and devices are shown next to each relevant content. In another interface an actual overlay icon is displayed where display order of these IoT devices are changed as the current viewing content changes (mostly by scrolling).

3. Mapper Module

This module tries to map the IoT device that the content is actually referring to, to one of the IoT devices in user IoT ecosystem. This module gives IoT abstraction to content, where the content need not be aware about an IoT ecosystem and can still provide recommendations. Mapping also involves conversion of commands and devices from the recommendation to an actual device and command.

4. IoT Ecosystem Discovery Module

This module does all the interactions with an IoT ecosystem provider which further can be a service or an application. This module is responsible for getting information, type of all the IoT devices that a user has and to communicate state changes to those IoT devices when a user interacts with an interface given by the overlay module.

IV. ALGORITHM

We follow some preprocessing steps to make the text ready for passing through the next step which uses Word embedding to process the preprocessed text and get probabilities of device and commands of those devices. The algorithm uses certain mapped keywords to represent each device and those keywords are arranged in a hierarchical order to get the probabilities of each IoT device.

A. Text Pre-Processing

The content gets passed through various steps to get the preprocessed text which results in more accurate IoT device and command prediction. The steps are as follows:

1. Removing capitalization in content text by replacing everything with lowercase variant.
2. Removing special characters which do not form a part of the word and extra whitespaces.
3. Removing stop words from the content.
4. Certain Parts of speech removal, only relevant parts of speech is retained in preprocessed text. Parts of speech like but not limited to: conjunctions, interjections, preposition, pronouns are removed.
5. Perform Vocabulary check, where in proper dictionary words are only taken forward.

After these preprocessing steps, word frequency calculation is performed, where count of each word appearing in the text is maintained (referenced as L_k). Vector representation of the words are taken from a pre-trained word embedding (like Google News Vec 300 negative word embedding). All these words together form the document T.

B. Device Selection through leveled Categories

Each IOT device is mapped under a category those IOT devices are represented or are used in, for example under home automation category devices like fridge, microwave are mapped. These categories will be represented as C_i where categories are denoted by i.

The categories are divided into levels, since various devices could be present under one category, and as the depth of the category level (j) increases a more accurate IoT device will be selected. C_i^j represents category i with level j.

Cosine Similarity (C.S.) then between this category word C_i^j and the word in the document (W_k) is calculated. Categories with similarity above a threshold ($> t_h$) are only retained thereafter for that particular word.

$$S_{i,k} = \text{Cosine Similarity}\ (C_i^j\ , W_k) \qquad (1)$$

$$S_{i,k} > t_h\ , \qquad (2)$$

For each word W_k similarity for all the categories at level j is calculated and threshold function applied. Top few (=s) filtered categories C_i^j for that particular word is retained only (denoted as C_s).

Devices then are calculated from these categories where the similarity index $S_{i,k}$ of each category is used as device representation index ${}_r^kD^i$ where ${}_r^kD$ represents devices available for word W_k

Category $S_{i,k}$ each has some devices under it ${}_r^iD$, and device relative similarity index is calculated and normalized through weighted average between 0 to 1.

For a word W_k, device probability is :

$$ {}_{r}^{k}D = \sum_i {}_{r}^{k}D^i \qquad (3)$$

Normalizing these similarity index to have a proper representation of probabilities

$$ {}_{r}^{k}D = {}_{r}^{k}D \;/ \sum_r {}_{r}^{k}D \qquad (4)$$

Each word W_K has some ${}_{r}^{k}D$ devices representing how probable is that word representing that device. After doing this for all words W_k in document T we have each word's most probable device predictions.

Now to calculate most probable device all of these words represent, probabilities for each device in each word are added and normalized.

$$d_r = \sum_k {}_{r}^{k}D \times L_k \qquad (5)$$

Since we also have count of each word L_k in D, we use weighted normalization to get the probabilities of the devices.

$$d_r = {\sum_k {}_{r}^{k}D \times L_k} \Big/ {\sum_r D_r} \qquad (6)$$

Now d_r represents the probability of a device representing document T.

If 2 or more than 2 device's probability difference is less than threshold t_r then this conflict is resolved by going deeper in the leveled categories (j + 1) and entire algorithm is repeated again. i.e. $Absolute(\, d_{r-1} - \, d_r) < \, t_r$, then Calculate for (j + 1) level.

C. Device Selection through web page type

Each visited webpage can be pre-mapped to a category (using an external service that gives relevant category of a given url and can be used as an accurate webpage category mapping). These categories are given probabilistic weights G_m and have a list of devices (H_m^i) each. The probabilistic weight G_m:

1) Represents the relevancy of the particular category in deciding an IOT device
2) Has equal distribution among possible devices.

G_m is used to first to boost certain devices already in our prediction list and then to introduce others devices that might not be represented by the content but by the application itself. The order is necessary to make sure our probability boosting technique is fair to devices already predicted d_r and the new devices H_m^i introduced, wherein the new devices introduced just because of this application will have lower probabilities.

$$d_r = \, d_r + \, H_m^r \, , \qquad (7)$$

So possible device set becomes:

$$\mathrm{D} = \{\mathrm{d}\!: \; d \; \in d_r \; or \; d \; \in \;\; H_m^i \, \} \qquad (8)$$

Normalizing the device probabilities

$$d_r = \, {d_r} \Big/ {\sum_r \, d_r} \qquad (9)$$

This gives us probabilities of IOT devices from T.

D. Device Command Prediction

We have already found out words W_r from T which fits or best represents each device. So only W_r is fed to this algorithm to avoid extra irrelevant computation.

Each device command is mapped to words that defines the action or words that represent objects that the device action needs, something like CAST action on a device could be represented by words like video (representing the object the action CAST needs).

Each device has actions A_r , each action a_r^u has a definition Q_r^u made by $q_v^{u,r}$ keywords. Vector of each $q_v^{u,r}$ represented by $b_v^{u,r}$ is pre-calculated and cached. Each $b_v^{u,r}$ then is compared with all the words W_r for device D_r.

$$s_{i,v}^{u,r} = \text{Cosine Similarity } (b_v^{u,r} \, , W_r^i \;) \qquad (10)$$

A threshold t_m acts as filter to let only relevant actions proceed.

$$s_{i,v}^{u,r} \; > \; t_m \, , \qquad (11)$$

$$S_u^r \; = \max(s_{i,v}^{u,r}) \qquad (12)$$

The device's action a_r^u probability is S_u^r defined as the max similarity index $s_{i,v}^{u,r}$ available.

Then S_u^r after normalization would represent the probability of that particular action on that device.

$$S_u^r = \; {S_u^r} \Big/ {\sum_u \, S_u^r} \qquad (13)$$

Similar approximate probability mapping for each device will be calculated and aggregated together to represent it in a meaningful way.

E. Optimizations

The algorithm can be further optimized by using additional methods and techniques. It is recommended to run the algorithm in parallel for different IoT device predictions. Running this algorithm only for IoT devices mapped to the category of current web page the user is consuming will further remove unwanted device computations.

A major optimization was found by arranging all category words in a single level i.e. C_i^j where j is constant in ascending order of the vector angle from any arbitrary vector. Such an ordering ensures that the computation of cosine similarities can be stopped in middle as cosine similarity will decrease after a given point, so whenever the cosine similarity goes below threshold, we can stop the computation for that particular device or command's probability.

Making sure that the Category words differ by a minimum of 0.09 threshold will also ensure that similar computations do not take place of an IoT devices. If 2 or more Category words are within 0.09 threshold limit for cosine similarity value, then chose a word which is farthest from the previous selected category word.). 0.09 was taken as a appropriate threshold limit after experimentations.

F. Custom IoT Tags Structure

The service "IoT device Parser" running on the cloud has "Tag Generation" module which is responsible for generating tags based on the predicted IoT devices and commands on those devices. We further define the tag structure has follows:

1) iot-device is the root tag for any device suggestion. This tag defines the iot device using attribute data-device-type, data-device-name and data-device-for. data-device-type should have a value from a standard set of values which is shared across systems so for similar devices have same data-device-type. data-device-for attribute has value of the unique selector of a parent root element so this tag can be separated from the root element which has the related content.

2) iot-command tag defines the iot command which can be operated on and needs a iot-device tag to ba a parent. This tag has attributes like data-iot-command-value-type, data-iot-command-pre, data-iot-command-post, data-iot-command-value. data-iot-command-value-type represents the command type in number, discreet value etc, this attribute defines the data type which should be used to associate values to the IoT command. Iot-command-pre and iot-command-post tags defines the pre and post commands that should be executed for a given command. data-iot-command-value attribute contains the actual value that should be used for the given IoT command as represented by iot-command tag.

V. SIMULATION AND RESULTS

This section covers the simulation environment used to verify the working, efficiency and feasibility of the proposed architecture and the proposed algorithm. The results obtained are from the simulation environment mentioned and strengthens the benefits obtained from following the proposed architecture.

A. Simulation

The simulation environment includes a hosted server in local network and mobile devices trying to interact with the local server. Since we couldn't control content providers directly, we made a proxy server which would fetch the webpages at URLs given by a database and then pass all of this content to the "IoT Device parser" Web Service and response from the same will be given to the handheld device. We also made a client side webpage which could fetch content in certain sections of the website from content in a database. For plugin check we used Google Chrome plugin development environment to develop and check workings of our plugin.

1) System Specs and setup

We have simulated the proposal with the below hardware setup:

- One Linux Machine acting as a Proxy Server
- Two Android Mobile devices (S8: Octa-core 2.3gHz Quad + 1.7 Ghz quad Exynos 9 Octa 8895 and Mali-G71 MP20GPU, 4GB LPDDR4 RAM) acting as web clients
- Some IoT devices like refrigerator, vacuum cleaner, T.V. etc. Although the findings in this paper can also be established without having these IoT devices.

This paper uses some open source tools and apps during simulation, some of the significant software's are listed below:

- Google Chrome Web browser with chromium-based rendering engine as web client
- Node.JS server on the proxy server
- Mongo NoSQL database for the Proxy Server
- SmartThings application for IoT ecosystem coverage and discovery.

B. Profiling

Our proposed architecture involves content-providers to either pass the rendered web-page to the web service or call this service from the client side.

1) When the "IoT device Parser" service was called from the server-side (passing it the HTML) it took around 217ms to process and send output to the server.

2) On client-side invocation of the service it takes 167ms to process and get the output.

3) The algorithm takes 136 ms to process and predict IoT devices when all the optimizations as mentioned are applied

C. Results

We have considered eleven commonly used IoT devices inside Smart Home. The devices include Television, Coffee Maker, Microwave Oven, Fridge, Speaker, and Washing Machine, Light bulb, Smartphone, Wearable Watches, Robot Vacuum Cleaner and Air purifier. For Content providers we have chosen nearly 1000 websites in different relevant domains from top 1 million Alexa websites [7]. These domains include entertainment, News, Weather, Health, Shopping, Music, food, business. We manually visited top 3 hyperlinks of these websites and marked the relevant devices and controls with respect to the content of these sites. Fig 6 represents the accuracy for the above experiment.

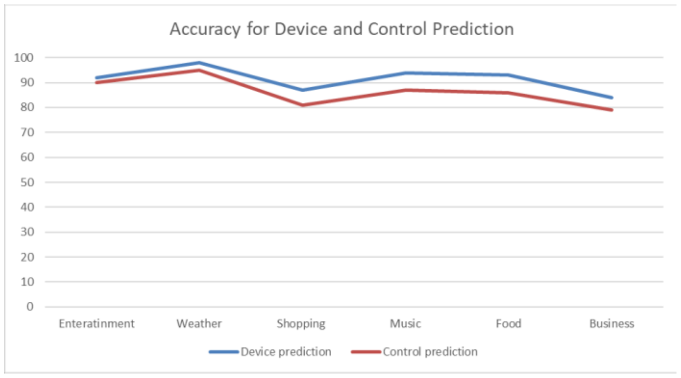


Figure 6 Accuracy for Device and Control prediction

Out of the 1000 web sites along with the hyperlinks visited, our algorithm predicted the relevant IoT devices from eleven commonly used devices. The predictions result shows more than 80% accuracy for most of the devices. [TABLE II].

TABLE II. WEB CONTENT TO DEVICE PREDICTION ACCURACY

Devices	**Web Pages Categories**	**Accuracy (%)**
Fridge	Food & Drinks, Storage	82.71
Microwave	Food, Cooking	79.91
Smart watch	Health, Exercise	87.25
Television	Education, Business, Finance, Entertainment, Sports, Music, Video	89.34
Air-Conditioner	Weather, Atmosphere, Temperature	90.18
Air Purifier	Weather, Air Ambience, Purify	81.37
Speaker	Audio Players, Entertainment, Sports, Music, Audio	78.56
Washing Machine	Clothes, Washing	78.54
Coffee Maker	Food & Drinks, Making	82.78
Robot Vacuum Cleaner	Robo, cleaning	79.89
Light Bulb	Bulb, Color, Light	81.16

Table 2. Web Content to Device Prediction Accuracy

VI. CONCLUSION

This papers proposed a mechanism by which the current available web content can be seamlessly converted into IoT enabled content. The proposed method also takes care of the privacy aspects of the user as the IoT device information is not exposed to the outside the Smart Home environment. The result shows we have decent accuracy in terms for devices and controls with respect to the relatable web contents from various Websites.

VII. FUTURE WORK

Although the proposed paper mainly centered on the web content, but this idea can be leveraged for native application contents and various data formats including audio, video and images. For example if the user watches a movie using the content recognition engine we can modify various IoT device controls including Light Setting, Air Conditioner, Blinds and other relatable devices. This proposed idea, if extended for many content types, will dramatically change the way the IoT devices are consumed inside the Smart Home environment.

VIII. REFERENCES

[1] Nay Min Htaik,Nyein Aye Maung Maung, and Win Zaw, “Enhanced IoT-based Interoperable and Configurable Middleware using Semantic Web Techniques,” ECTI-CON, 2018.

[2] Daoqu Geng, and Qi Gao,”Semantic Web Technology and Prolog Reasoning Based Task Planning Mechanism and Application in Smart Home for Internet of Things,” ICEIEC,2019.

[3] Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello,”A Semantic Web Approach to Simplifying Trigger-Action Programming in the IoT”. 2017.

[4] Agnese Chiatti, Mu Jung Cho. Text Extraction and Retrieval from Smartphone Screenshots: Building a Repository for Life in Media. SAC '18 Proceedings of the 33rd Annual ACM Symposium on Applied Computing Pages 948-955.

[5] Joulin, Armand, et al. "Fasttext. zip: Compressing text classification models." arXiv preprint arXiv:1612.03651 (2016).

[6] Ankit Jain, Om Prakash, Arka talukdar, Siba Prasad Samal, Ravi Nanjundappa, and Mahesha N, "Screen Intelligence Engine: ML based method for predicting IoT devices and actions using screen contents," unpublished

[7] https://www.alexa.com/topsites

[8] Moataz Soliman,Tobi Abiodun, Tarek Hamouda,Jiehan Zhou, and Chung-Horng Lung, "Smart Home: Integrating Internet of Things with Web Services and Cloud Computing ," 5th International Conference on Cloud Computing Technology and Science, 2013

[9] Feng-Long Huang, Sheng-Yi Tseng, "Predictable smart home system integrated with heterogeneous network and cloud computing", Machine Learning and Cybernetics (ICMLC) 2016 International Conference on, vol. 2, pp. 649-653, 2016.

[10] https://developers.google.com/web/fundamentals/web-components/customelements

[11] Tahar Dahoumane,Mourad Haddadi,Zouhir Amokrane, "Web Services and GSM based Smart Home Control System,"International Conference on Applied Smart Systems (ICASS), 2018