


Article

Toward Flexible and Efficient Home Context Sensing: Capability Evaluation and Verification of Image-Based Cognitive APIs [†]

Sinan Chen ^{1,*} , Sachio Saiki ¹ and Masahide Nakamura ^{1,2}

¹ Graduate School of System Informatics, Kobe University, 1-1 Rokkodai-cho, Nada, Kobe 657-8501, Japan; E-Mails: sachio@carp.kobe-u.ac.jp (S.S.); masa-n@cs.kobe-u.ac.jp (M.N.)

² RIKEN Center for Advanced Intelligence Project, 1-4-1 Nihonbashi, Chuo-ku, Tokyo 103-0027, Japan

* Correspondence: chensinan@ws.cs.kobe-u.ac.jp; Tel.: +81-78-803-6295

[†] This paper is an extended version of the conference paper: Sinan, C.; Sachio, S.; Masahide, N. Evaluating Feasibility of Image-Based Cognitive APIs for Home Context Sensing. In proceedings of the ICSPIS 2018, Dubai, UAE, 7–8 November 2018.

Received: 26 December 2019; Accepted: 2 March 2020; Published: 6 March 2020



Abstract: Cognitive Application Program Interface (API) is an API of emerging artificial intelligence (AI)-based cloud services, which extracts various contextual information from non-numerical multimedia data including image and audio. Our interest is to apply image-based cognitive APIs to implement flexible and efficient context sensing services in a smart home. In the existing approach with machine learning by us, with the complexity of recognition object and the number of the defined contexts increases by users, it still requires directly manually labeling a moderate scale of data for training and continually try to calling multiple cognitive APIs for feature extraction. In this paper, we propose a novel method that uses a small scale of labeled data to evaluate the capability of cognitive APIs in advance, before training features of the APIs with machine learning, for the flexible and efficient home context sensing. In the proposed method, we exploit document similarity measures and the concepts (i.e., internal cohesion and external isolation) integrate into clustering results, to see how the capability of different cognitive APIs for recognizing each context. By selecting the cognitive APIs that relatively adapt to the defined contexts and data based on the evaluation results, we have achieved the flexible integration and efficient process of cognitive APIs for home context sensing.

Keywords: smart home; contexts; cognitive API; image; document similarity; internal cohesion; external isolation; clustering

1. Introduction

With the rapid progress of ICT and Internet of Things (IoT) technologies, research and development of smart homes have been actively conducted. In smart homes, it is common to use ambient and/or wearable sensors such as temperature, humidity, motion, and accelerometer in order to retrieve contexts of users and homes for achieving various value-added services such as References [1–3]. In recent years, Artificial Intelligence (AI) and cloud computing technologies have brought enormous development potentiality for smart home services. With the progress of emerging deep learning, retrieving the informative features of home contexts is not limited to conventional sensors data [4,5], but includes multimedia data such as the research in References [6–8]. Using multimedia data, such as image and audio, for home context sensing is promising for value-added smart services, since the multimedia data contain richer information than the conventional sensor data. However, recognizing multimedia data generally requires massive computation. It was thus unrealistic for general households to install and maintain such a complex and tedious system at

home. In recent years, a cognitive service provides the capability to understand multimedia data based on sophisticated machine-learning algorithms powered by big data and large-scale computing resources. Typical services include image recognition [9], speech recognition [10], and natural language processing [11]. A cognitive service usually provides cognitive APIs (Application Program Interface), with which developers can easily integrate powerful recognition features in their own applications. We consider that cognitive APIs make full use of multimedia data. Therefore, they have great potential to improve smart homes since the user would no longer need to maintain a complex and tedious system. Although various kinds of cognitive APIs exist and have been researching such as [12–14], we especially focus on image recognition APIs in this paper. An image recognition API receives an image from an external application, extracts specific information from the image, and returns the information as a set of words called tags. The information of interest varies between services. For example, Microsoft Azure Face API [15] estimates age, sex, and emotional values from a given human face image. IBM Watson Visual Recognition [16] recognize items in the image such as home appliances, furniture, and tools.

The main contribution of this paper is to propose a novel method that uses a small scale of labeled data, to evaluate the capability of cognitive APIs in advance, before training features of the APIs with machine learning, for the flexible and efficient home context sensing. We generally divide the t (the total number of labeled data) into three levels: (1) large scale ($t > 10000$), (2) moderate scale ($10000 \geq t \geq 1000$), (3) small scale ($t \leq 100$). In the existing home context sensing approach with machine learning [17,18], with the complexity of recognition object and the number of the defined contexts increase by users, it still requires directly manually labeling a moderate scale of data for training and continually try to calling multiple cognitive APIs for feature extraction. However, for each defined home context, according to different capabilities by cognitive APIs, there will be a difference among difficult-to-train data. That is, the individual contexts with low recognition accuracy by the different APIs, which requires us to make a capability evaluation of each cognitive API and defined context in advance before manually labeling a lot of data. Using the proposed method, one can understand the coverage and limitation of different APIs towards specific home contexts for flexible integration. Also, one can reduce unnecessary data manual labeling and calling cognitive APIs for an efficient process.

The previous version of this paper was published as a conference paper [19]. Changes made to this version are most significantly the addition of clustering algorithms and the model construction based on the former. In the proposed method, we first capture images of different contexts. Afterward, we send the image to the cognitive API to retrieve tags from the images. Finally, integrating the internal cohesion and external isolation concepts into the clustering results, we evaluate the capability of the APIs by checking if the tags can sufficiently characterize (or distinguish) the context shown in the original image. Our key idea of evaluation is to integrate document similarity measures [20–22], the concepts (i.e., internal cohesion and external isolation [23–26]) into results of clustering [27,28], to see how the capability of different cognitive APIs for recognizing each context. More specifically, we evaluate the clustering algorithm results, with respect to the internal cohesion and external isolation. That is, we see if cluster belonging to the same (or different) context(s) are associated with similar tags (or dissimilar tags, respectively). Based on the evaluation results, we produced a flexible and efficient way to select the high capability APIs for building a high accuracy model with machine learning.

Based on the proposed method, we have experimented with the smart home space of our laboratory. Follow the proposed steps, we have completed the capability evaluation and verification of cognitive APIs. The experimental results showed that the five of seven contexts recognition accuracy reached 100%, the remaining difficult contexts also well within the response range of evaluated results. This fully shows the reliability of our proposed method. The remainder of this paper is organized as follows. Section 2 introduces the related work of cognitive APIs from recent years. Section 3 produces a complete description of the proposed method. The experimental evaluation

and verification of image-based cognitive APIs for home context sensing are presented in Section 4, followed by conclusions in Section 5.

2. Related Work

To retrieve contexts of users and homes for achieving various value-added services, simplify the process and improve the quality of the original is significant. As described in the introduction, with the kind and number of the home context increase from one house to another, it is always a key difficult issue to simplify the model and achieve a more efficient process. In this section, we introduce some related works in the smart home field from recent years around the above issues.

Tax et al. [29] provide a novel algorithm to extract those relevant parts of the data for support counting, only consider specific parts of the datasets instead of the full dataset, which allows speed up the counting of the support of a pattern. Unlike their approach, we use all dataset made by labeling selected by manually that applies into machine learning algorithms. The core of this paper is to present a method that by evaluating, in order to help the user to know how to select original data better in advance. The research in Reference [30] proposes a distributed service-oriented architecture (D-SOA) for a smart-home system, which improved communication efficiency, reduction in network load, and response time. Comparing with their approach, we got the same achievements, by reducing the process to call the low-capability APIs for specific contexts. Xu et al. [31] proposed the software-defined smart home platform, which flexibly adapts to the great difference between family scenes and user demands. We are also focusing on this key point. Unlike their research, we use a fixed-point camera rather than smart devices with an interface for our study, in order to reduce the complexity of system operation by the user. However, for the implementation, the number and setting position of the camera is a big issue. The research in Reference [32] concerned with the use of traffic classification techniques for inferring events taking place within a building, in order to improve security and privacy concerns of the smart homes. In our study, the security and privacy all depend on each cloud service environment of cognitive APIs. Due to the local system does not save any images and information of users, to reduce the unnecessary APIs calling can significantly improve this issue by the proposed method in this paper. Stojkoska et al. [33] present a three-tier Internet of Thing based hierarchical framework for the smart home using fog computing. Although local computation is cheaper operation than communication, it is difficult to retrieve rich and full feature values for fine-grained home context sensing in a short time. It might make simple problems become complicated. To simplify the process of using cloud computing, and improve the quality of the labeled data is the core of this paper.

3. Methodology

This section describes the previous method of this paper [19], emphatically presents the proposed method in this paper, and discusses the related techniques.

3.1. Previous Method

Since the existing APIs are trained for general-purpose image recognition, they may not be of practical use in the specific configuration of smart homes. In the previous version of this paper, we presented a method that evaluates and compares the capability of multiple image recognition APIs using a few image data, for a given set of home contexts. Figure 1 depicts the essential part of the previous method. In the figure, $\{c_1, c_2, \dots, c_m\}$ represent a given set of home contexts. For each context, we collect n images at home, then send the images to cognitive APIs. Finally, we evaluate the performance of the APIs, by analyzing the output tags. More specifically, the previous method consists of the following five steps:

Step 1: Acquiring images

A user of the proposed method deploys an image capturing device (e.g., USB camera) in the target space, and configures the device to take snapshots of the space periodically with an appropriate interval.

Step 2: Defining home contexts to recognize

The user defines a set $C = \{c_1, c_2, \dots, c_m\}$ of home contexts to be recognized by the cognitive API, such as “Dining”, “Cleaning”, “Nobody” and so on.

Step 3: Selecting representative images

For each context $c_i \in C$, the user manually selects representative n images $IMG(c_i) = \{img_{i1}, img_{i2}, \dots, img_{in}\}$ that well expose c_i , from all images obtained in Step 1. Note, the user needs to select images on different days as possible, to avoid overfitting the data. To evaluate the capability of cognitive APIs, the user can first select a small scale of image data (Generally, $n \leq 10$ for keeping $m \times n$ with a small scale range). The specific number depends on the situation.

Step 4: Calling cognitive API

The user designates a set $API = \{api_1, api_2, \dots, api_q\}$ of cognitive APIs to be evaluated. There are many cognitive APIs that extract tags from images. For every $c_i \in C$, $img_{ij} \in IMG(c_i)$, and $api_k \in API$, $api_k(img_{ij})$ is invoked, and a set $Tag(img_{ij}, api_k) = \{w_1, w_2, w_3, \dots\}$ of output tags is obtained. $Tag(img_{ij}, api_k)$ represents a recognition result for cognitive API api_k for an image img_{ij} belonging to a context c_i . The size of $Tag(img_{ij}, api_k)$ varies for img_{ij} and api_k . Since there are m contexts, n images for each context, and q APIs, this step creates totally $m \times n \times q$ sets of output tags.

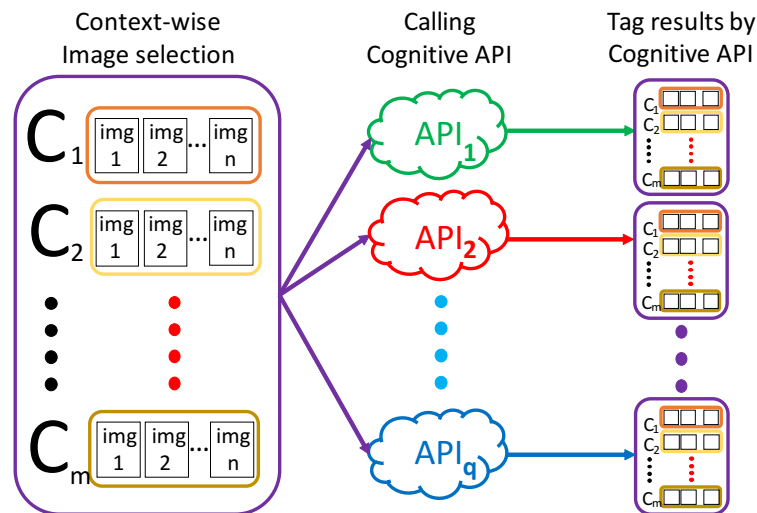


Figure 1. The flow from context label setting to analysis of results.

Step 5: Analyzing output tags

Step 5-1: Encoding output tags

The vector is a numerical representation of the document, where each component of the vector refers to a tag. It shows the presence or importance of that tag in the document. More specifically, regarding every set $Tag(img_{ij}, api_k)$ of output tags as a document corpus, the user can extract features from each document, by converting every set $Tag(img_{ij}, api_k)$ into a document vector $V(img_{ij}, api_k) = \{v_1, v_2, \dots\}$, using a document vectorizing technique, such as *TF-IDF* [34], *Word2Vec* [35], *Doc2Vec* [36], *GloVe* [37], *fastText* [38] and so on. Listing 1 shows an example of encoding output tags by the TF-IDF method with python.

Step 5-2: Document similarity measure

Regarding each document vector in $\bigcup_{ij} V(img_{ij})$ of each api_k , the method calculates the similarity or distance, which is denoted as ' \approx ', between any two of documents using a certain method of the document similarity measure. Regarding the calculation of *document similarity*, there exists a variety of methods in the field of natural language processing, such as *Cosine Similarity* [39], *Euclidean Distance* [40], *Pearson Correlation Coefficient* [41] and so on.

Step 5-3: Analyzing document similarity

For each api_k , the user evaluates the performance of api_k of context recognition, with respect to internal cohesion and external isolation. The internal cohesion represents a capability that api_k can produce similar output tags for images in the same context. That is, for $c_i \in C$, we evaluate $Tag(img_{ij}, api_k) \approx Tag(img_{ij'}, api_k)$. On the other hand, the external isolation represents a capability that api_k can produce dissimilar output tags for images in different contexts. That is, for $c_x \neq c_y$, we evaluate $Tag(img_{xj}, api_k) \not\approx Tag(img_{yj'}, api_k)$.

Listing 1: Example of encoding output tags by the TF-IDF method with python

```
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np
import pandas as pd

for api in ["API_name"]:
    tags = np.array(tags_labels_pd[api])
    contexts = np.array(tags_labels_pd["labels"])
    vectorizer = TfidfVectorizer(use_idf=True)
    vecs_tfidf = vectorizer.fit_transform(tags)
    np.set_printoptions(precision=3)
    np.set_printoptions(threshold=np.inf)
    tfidf_vectors = vecs_tfidf.toarray()
    feature_names = vectorizer.get_feature_names()
    feature_names = list(feature_names)
    feature_names.append("labels")

    vectors_pd = pd.DataFrame(tfidf_vectors)
    vectors_pd = vectors_pd.round(3)

    labels_pd = pd.DataFrame(contexts)
    vectors_labels_pd = pd.concat([vectors_pd, labels_pd], axis=1)

    vectors_labels_pd.columns = feature_names

vectors_labels_pd
```

3.2. Proposed Method

As follow-up studies [17,18] continue, our work is not limited to simply evaluating the capability of cognitive APIs, but more focus on the implementation of a flexible and efficient process for home context sensing. As we mentioned in Section 1, we are struggling to understand the coverage and limitation of different APIs towards specific home contexts, and reduce unnecessary data manual labeling and calling cognitive APIs process. Based on Step 1 to Step 5-1 in the previous method, changes made to this version are most significantly the addition of the new Step 5-2 and 5-3, for evaluating the capability of cognitive APIs, and presents the Step 6-1 to Step 6-4 for building model to verifying the evaluation results. The following explains the proposed new steps in detail.

Step 5: Analyzing output tags**New Step 5-2: Clustering document vectors**

The user first randomizes the order of all document vectors $\cup_{ij} V(img_{ij})$ of each api_k along with the corresponding labels, then split them into the non-labeled document vectors and the known labels. After that, the user applies a clustering algorithm W into the randomized non-labeled document vectors of each api_k . The algorithm W include k -means [27], Partitioning Around Medoids (PAM) [42], Clustering Large Applications (CLARA) [43] and so on. This step is a process

of automatic classification, which requires the user to define the number of clusters to classify in advance. By using the clustering algorithm, it returns the integer labels corresponding to the different clusters in the results. Listing 2 shows an example of applying k-means++ algorithm into document vectors with python. Further more, an example of the clustering results in cross-tab is shown in Figure 2a, which produced by integrating the known labels and the returned integer labels. The evaluation key is the number of all labels more concentrated in the different classes of both rows and columns, the better the classification effect. Since the naive clustering algorithm cannot show the classification effect well to a small scale of data in general, the new Step 5-3 improves it.

Listing 2: Example of applying k-means++ algorithm into document vectors with python

```
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.cluster import kmeans
import numpy as np
import pandas as pd

random_vectors_labels_pd = vectors_labels_pd.sample(frac=1)
random_vectors_np = random_vectors_labels_pd.iloc[:, :-1].values
random_labels_np = random_vectors_labels_pd.iloc[:, -1:].values

kmeans.euclidean_distances = cosine_similarity
model = kmeans(n_clusters=len(sorted(list(set(random_labels_np)))) ,
               init='k-means++')
model_output_labels_np =
    model.fit_predict(non_labels_random_vectors_pd)

model_output_labels_pd = pd.DataFrame(model_output_labels_np ,
                                     columns=[ 'Assignments' ])
random_labels_pd = pd.DataFrame(random_labels_np , columns=[ 'labels' ])

labels_concat_pd = pd.concat([ random_labels_pd ,
                               model_output_labels_pd ], axis=1)
result_crosstab_pd = pd.crosstab(labels_concat_pd[ 'assignments' ] ,
                                 labels_concat_pd[ 'labels' ])

result_crosstab_pd
```

New Step 5-3: Analyzing clustering results

From the results of automatic classification of each api_k , the user evaluates the recognition capability of c_i ($c_i \in C$). The core of the evaluation method is to integrate the internal cohesion and external isolation concepts into the clustering results. Specifically, Listing 3 shows an example of the key method for evaluating the capability of cognitive APIs with python. Further more, an example of the process for evaluating the capability of cognitive APIs shown in Figure 2, including a calculation formula and the principles. As the evaluation description of this step, refer to the cross-tab of Figure 2b, the capability evaluation method includes two points:

- (1) The maximum value in each row shows the capability of api_k for the row c_i . Note, it cannot conduct the evaluation using this score if no maximum value in that row.
- (2) The more there are other values in the row or column of the maximum value of each row, the lower the capability of api_k for c_i of the row where the maximum value is.

Here, c_d ($c_d \in C$) with low scores may be regarded as difficult-to-train contexts.

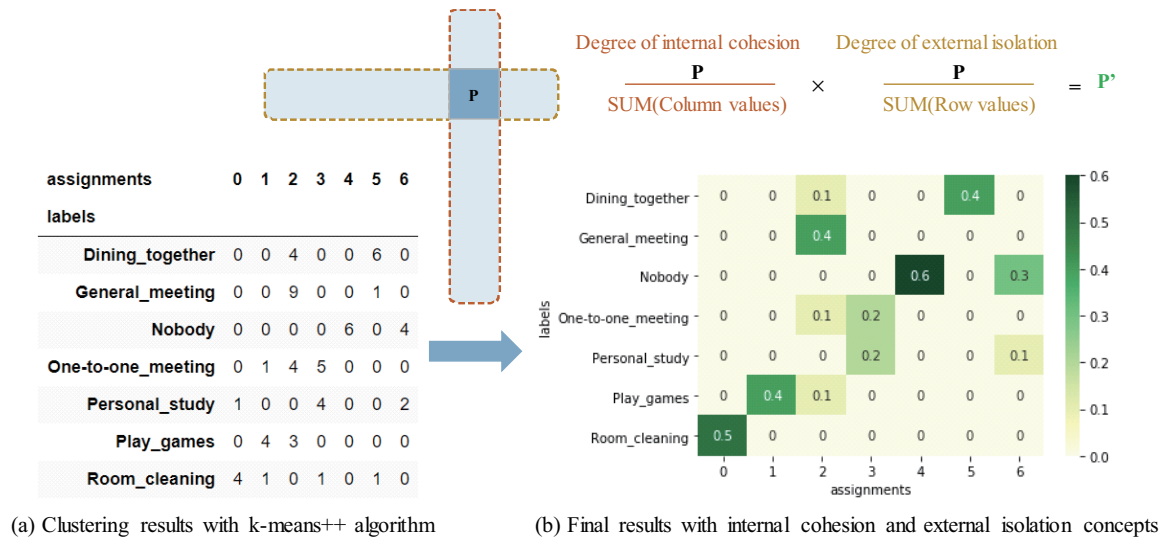


Figure 2. Example of the process for evaluating the capability of cognitive Application Program Interfaces (APIs).

Listing 3: Example of the key method for evaluating the capability of cognitive APIs with python

```
import numpy as np
import pandas as pd
import seaborn as sns

sum_row_values = result_crosstab_pd.sum(axis=1)
sum_column_values = result_crosstab_pd.sum(axis=0)

final_results_list = []
for i in range(len(result_crosstab_pd)):
    this_row_results = []
    this_row = result_crosstab_pd.iloc[i:i+1,:]
    sum_this_row_values = float(sum_row_values[i])
    for j in range(len(result_crosstab_pd)):
        this_value = this_row.iloc[:,j:j+1]
        this_value = float(this_value.values)
        sum_this_column_values = float(sum_column_values[j])
        if this_value != 0:
            this_value = ((this_value/sum_this_row_values)
                          * (this_value/sum_this_column_values))
        else: pass
    this_row_results.append(round(this_value,1))
    final_results_list.append(this_row_results)
final_results_pd = pd.DataFrame(final_results_list,
                                index=result_crosstab_pd.index,
                                columns=result_crosstab_pd.columns)

sns.load_dataset('iris')
plt.figure()
sns.heatmap(final_results_pd, cmap="YlGn", annot=True, cbar=True)
```

Step 6: Selectively building Model

In this Step, based on the evaluation results in Step 5, the user can select high-capability APIs, for improving the difficult-to-train contexts in the built model, and produce an efficient process.

Step 6-1: Preparing labeled image data

Follow Step 1 to Step 3 in Section 3.1, the user collects and labels a moderate scale of image data required for building a model with machine learning (Generally, $n' \geq 10n$). The user also selects the original image data with more prominent features for the previously known difficult-to-training contexts c_d ($c_d \in C$), which may bring a buffer value to the accuracy of other contexts.

Step 6-2: Selecting high-capability APIs to call

As a addition in Step 4 in Section 3.1, the user first determine the difficult-to-train contexts c_d , and to select multiple high capability $API_h = \{api_1, api_2, ..., api_g\}$ ($API_h \in API, g < p$) from evaluation results of the new Step 5-3. The selection approach as follows.

(1) The user can select $API_h(g = 1)$ that with maximum total scores of contexts, to ensure the built model with high overall accuracy.

(2) The user can also select $API_h(g = 2)$ with high complementarity of context evaluation results, to ensure the built model with high average accuracy.

The approach (2) for home context sensing in most cases better than the (1), because of too much or too little will have an effect on the accuracy, efficiency, and complexity. Therefore, to select the small number of high-capability APIs rather than reduce the dimension of features can produce more applicable features related to c_i , for improving the process efficiency.

Then, for every $c_i \in C$, $img_{ij} \in IMG(c_i)$, and $api_{k'} \in API_g$, $api_{k'}(img_{ij})$ is invoked, and a set $Tag(img_{ij}, api_{k'}) = \{w_1, w_2, w_3, ...\}$ of output tags is obtained. $Tag(img_{ij}, api_{k'})$ represents a recognition result for $api_{k'}$ for an image img_{ij} belonging to a context c_i . The size of $Tag(img_{ij}, api_{k'})$ varies for img_{ij} and $api_{k'}$. Since there are m contexts, n' images for each context, and g APIs, this step creates totally $m \times n' \times g$ sets of output tags.

Step 6-3: Encoding and combining features

Follow Step 5-1 in Section 3.1, regarding every set $Tag(img_{ij}, api_{k'})$ of output tags as a document corpus, the user can first extract features from each document, by converting every set $Tag(img_{ij}, api_{k'})$ into a document vector $V(img_{ij}, api_{k'}) = \{v_1, v_2, ...\}$ (see Listing 1). Then, for each img_{ij} , the user combines all $V(api_{k'})$ into $\bigcup V(img_{ij}, api_{k'})$.

Certainly, the another approach is to combine all $Tag(img_{ij}, api_{k'})$ into $\bigcup Tag(img_{ij}, api_{k'})$ in first, then converting them to document vectors. However, in this way, it could have influenced encoding results if there are the same tags output by API_h . We do not think that it means the features are highlighted in the related img_{ij} .

Step 6-4: Building a model and Verifying results

For each context c_i , the user split $\bigcup V(img_{ij}, api_{k'})$ into training data and test data. The user first applies a supervised machine learning algorithm A into the training data and corresponding labels for building a model M . The algorithm A include *Support Vector Machine (SVM)* [44], *Neural Network (NN)* [45], *Decision Tree* [46] and so on. Then, using the test data, the user evaluates the built model M , by checking the output c_i with the labeled test(c_i). The more M outputs the correct contexts, the M is more accurate.

As the verification approach, the user can verify if the capability of the selected APIs the same as expected, by checking the accuracy of overall, average, context-wise and so on. The user can also check if the difficult-to-training contexts c_d evaluated from the new Step 5-3 indeed difficult to recognize, and if the accuracy of them is improved.

4. Experimental Evaluation and Verification

This section introduces an experiment conducted for evaluating the capability of cognitive APIs, and verifying the evaluation results, to produce a flexible and efficient process of home context sensing.

4.1. Experimental Setup

In this experiment, we set the target space to be a smart home space, which is a part of our laboratory. For Step 1, we install a USB camera to acquire images of the daily activities of members of

the laboratory. We develop a program that takes a snapshot with the USB camera every five seconds, and the images are cumulated in a server for nine months. In Step 2, we define seven contexts: “Dining together”, “General meeting”, “Nobody”, “One-to-one meeting”, “Personal study”, “Play games”, and “Room cleaning”. The representative images of each context and USB camera in this experiment is shown in Figure 3.

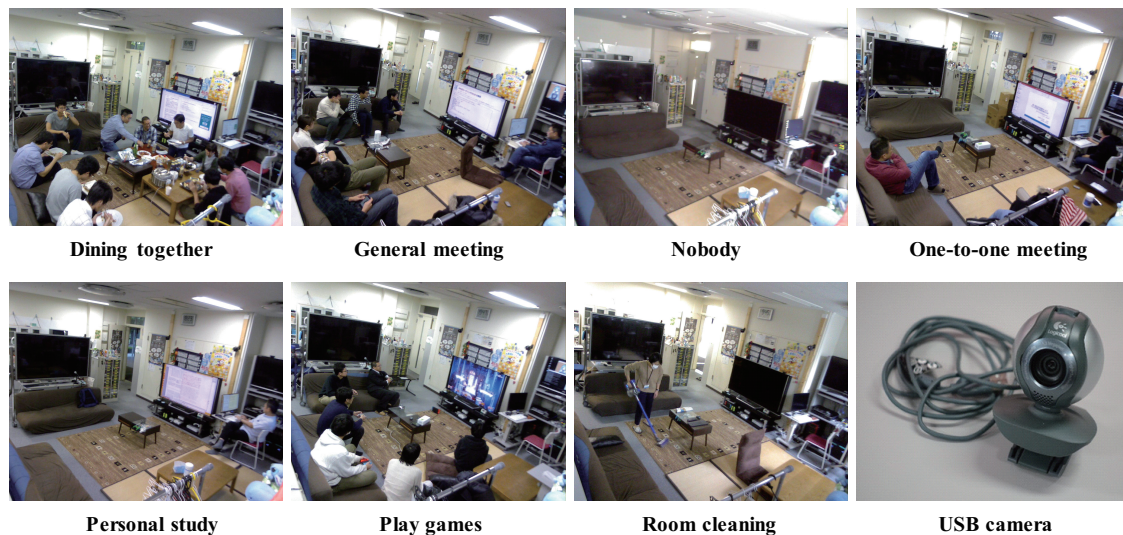


Figure 3. The representative images of each context and USB camera in this experiment.

4.2. Evaluating Capability of Cognitive APIs

In Step 3, for each context, we selected 10 representative images considered to expose the context well. The selection is done by visual inspection so that the 10 images are chosen from different dates and times as possible. In Step 4, the images are sent to the five different APIs: Microsoft Azure Computer Vision (Azure) API [47], IBM Watson Visual Recognition (Watson) [16], Clarifai API [48], Imagga REST (Imagga) API [49], and ParallelDots API [50]. The total 350 sets of output tags ($= 7 \text{ contexts} \times 10 \text{ images} \times 5 \text{ APIs}$) are obtained. The Step 5-1 to Step 5-3 have conducted in python by us (see Listing 1, Listing 2, and Listing 3). In the new Step 5-1, we used Term Frequency - Inverse Document Frequency (TF-IDF) [34] to encode each set of output tags to a vector. In the new Step 5-2, we applied all document vectors into cosine similarity [21] and k-means++ [51] algorithms, producing a process of automatic classification. In the new Step 5-3, as the capability evaluation, for the clustering results of each API, we calculated the scores of each context with internal cohesion and external isolation concepts into the cross table. We also calculated the total score of each context and API for making a more detailed analysis in the table.

4.3. Building a Model to Verify

Based on the evaluation results of the new Step 5-3, we implemented to selectively building model. In Step 6-1, follow Step 1 to Step 3, we selected 100 representative images for each context. Especially, we selected the original images of the difficult-to-train context with more prominent features. In Step 6-2, we selected two APIs with most high-capability (i.e., Clarifai API and Imagga API), and respectively sent the representative images to them for obtaining the output tags. In Step 6-3, we respectively encoded the output tags to document vectors with the TF-IDF method, and combined all document vectors for each image. In Step 6-4, for each context, we split all document vectors into half, as training data and test data. We first applied the Multi-class Neural Network algorithm into the training data using Microsoft Azure Machine Learning [52]. Then, we using the test data to evaluating the recognition accuracy of the build model. We compared the capability of the selected high-capability APIs, and the accuracy of difficult-to-training contexts, with the evaluation results of the new Step 5-3.

4.4. Results

Figure 4 shows the capability evaluation results of five cognitive APIs in this experiment. Among the five evaluation results, the contexts that with relatively good stability include the results in “Play games” of Watson API, “Dining together” of Imagga API, and “Nobody” of Paralleldots. Because in the above results that other values are not in the row or column of the maximum value of that row. In contrast, the contexts that with relatively bad stability (i.e., difficult-to-training contexts) include the results in “Play games” of Clarifai API and ParallelDots API, “Room Cleaning” of Watson API, Clarifai API, and Imagga API. Table 1 shows the maximum value in each row of the capability evaluation results of each API from Figure 4. From the total score of each API, the APIs that with relatively good capability include Imagga API and Clarifai API. In contrast, the APIs that with relatively bad capability include Watson API and ParallelDots API. From the total score of each context, the contexts that easily to be recognized include “Nobody” and “General meeting”. In contrast, the contexts that difficult to be recognized include “Room cleaning” and “Play games”. Figure 5 shows the results by combining features of the selected APIs (Clarifai and Imagga APIs). From the main results of the metrics, the overall accuracy reached around 0.977, and the average accuracy reached around 0.993. From the results of the confusion matrix, the contexts that with accuracy reached 100% include “Dining together”, “General meeting”, “Nobody”, “One-to-one meeting”, and “Personal study”. The accuracy of “Play games” was 96.1%, and “Room Cleaning” was 88.2%. The result of the built model reached the same as we expected. Especially, it verified that the difficult-to-train contexts evaluated from the new Step 5-3 indeed difficult to recognize, but the accuracy of them to a large extent was improved.

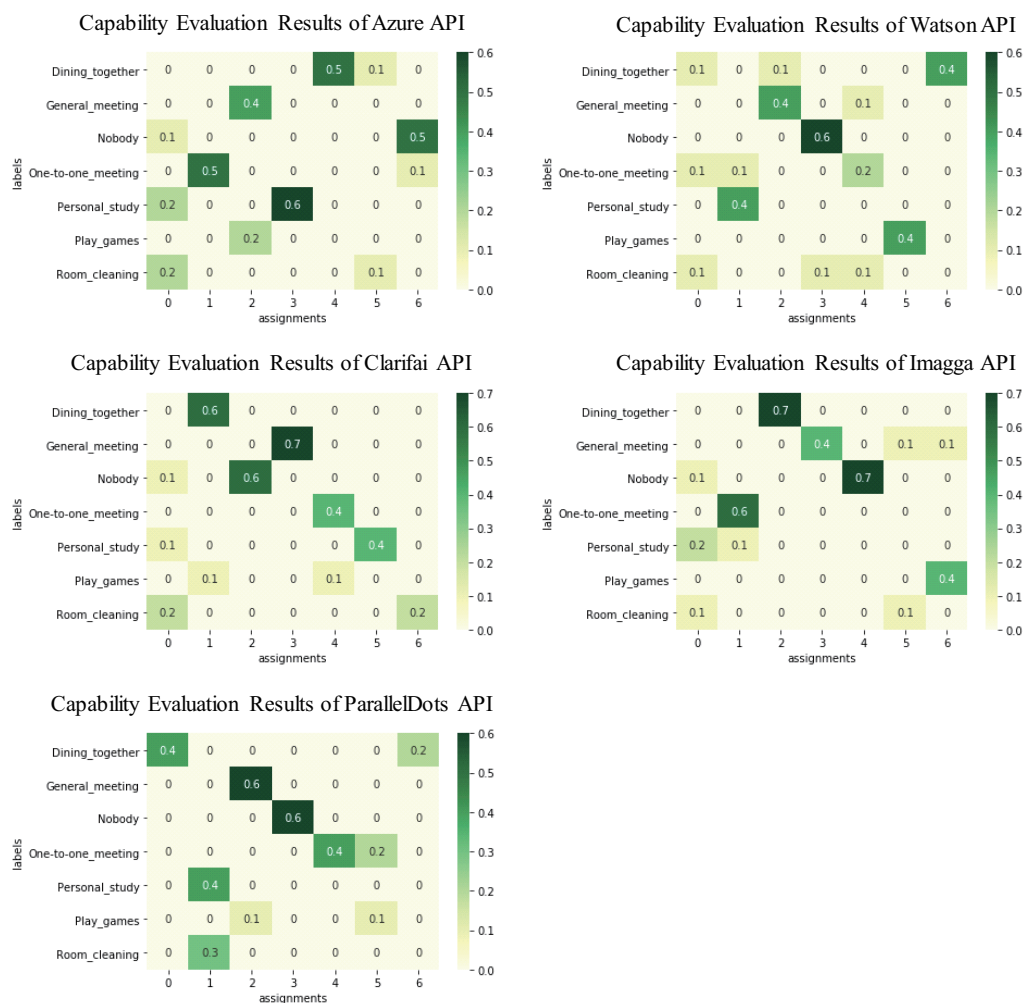


Figure 4. The capability evaluation results of five cognitive APIs in this experiment.

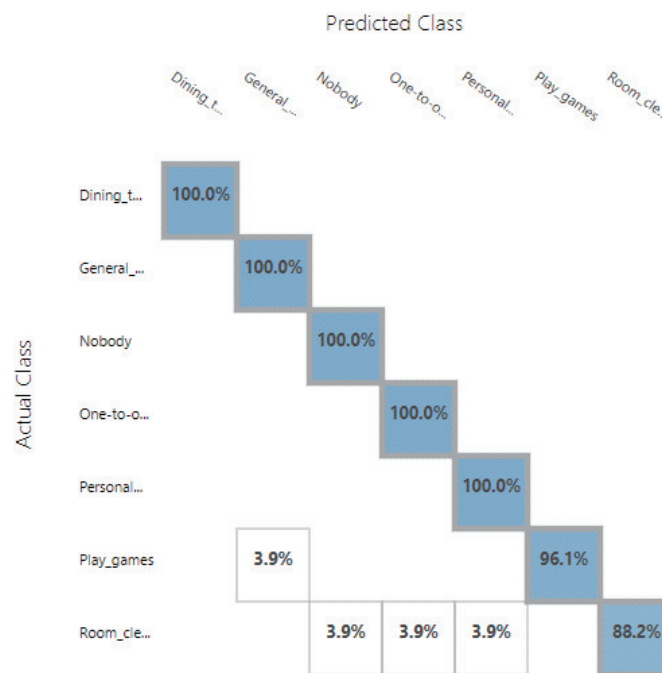
Table 1. The maximum value in each row of the capability evaluation results of each API from Figure 4.

Context Names	Azure API	Watson API	Clarifai API	Imagga API	ParallelDots API	Total
Dining together	0.5	0.4	0.6	0.7	0.4	2.6
General meeting	0.4	0.4	0.7	0.4	0.6	2.5
Nobody	0.5	0.6	0.6	0.7	0.6	3
One-to-one meeting	0.5	0.2	0.4	0.6	0.4	2.1
Personal study	0.6	0.4	0.4	0.2	0.4	2
Play games	0.2	0.4	0.1	0.4	0.1	1.2
Room cleaning	0.2	0.1	0.2	0.1	0.3	0.9
Total	2.9	2.5	3	3.1	2.8	

Metrics

Overall accuracy	0.977143
Average accuracy	0.993469
Micro-averaged precision	0.977143
Macro-averaged precision	0.978268
Micro-averaged recall	0.977143
Macro-averaged recall	0.977591

Confusion Matrix

**Figure 5.** The results by combining features of the selected APIs (Clarifai and Imagga).

4.5. Discussion

In this study, the factors that influence the home context sensing were many, which required to be considered. More specifically, the main factors in the different home contexts: (1) the position-change-degree of persons. (2) the number of persons existing. (3) the richness degree of objects existing. The contexts defined by us in this experiment covered the difference of the above (1) (2) (3) cases. In this way, we easily understand that the recognition accuracy of “Room cleaning” was not good in Figure 5, because of the (1) (2). A snapshot only represents the contents in a moment, which cannot retrieve more informative features on the time series. For another difficult home context “Play games”, the difference was existing among the labeled data due to the above (2). In the play

games every time, the number of persons with the difference between two to five. Further more, in the results of Figure 4, we regard the maximum value of each row as the easy degree of that context to be recognized. We have put them into Table 1 for easier to check. However, as Figure 4 shows, the other values in the row or column of the maximum value of each row still inevitable in most situations. They reflected the disturbance items for each context were existing, which should be also considered in the final evaluation calculation. Such as letting the maximum values subtract the other values that in the same row and columns, it might a good way.

5. Conclusions

In this paper, a method that uses a small scale of labeled data to evaluate the capability of image-based cognitive APIs in advance, towards the flexible and efficient home context sensing, is proposed. From experimental evaluation and verification, the high-capability APIs and difficult-to-train contexts well within the response range of evaluated results, confirming the advantage that the predictability and efficiency of feature extraction with cognitive APIs are improved by the proposed method.

The initial thinking of our study is to realize a system, where a simple edge system just capturing, and pre-processing images are deployed at home. All heavy tasks of image recognition are delegated to the cognitive service in the cloud. However, in the complex and different environments by one household to another household, for the APIs with many difficult-to-training data, it is still a big challenge that, how to rapidly and accurately obtain the more fine-grained evaluation results in advance. We have tried some experiments to extract more valuable features from the return results of each API, such as the score or confidence value of each output tag. However, the way has not been found to utilize them, due to there existing the big difference in the range and distribution of that score values by the different APIs. As future work, we will try to find the edge computing techniques with image recognition for implementing a more smart home context sensing.

Author Contributions: Writing—original draft preparation, S.C.; writing—review and editing, S.C. and S.S.; supervision, M.N.; validation, S.C.

Funding: This research received no external funding.

Acknowledgments: This research was partially supported by JSPS KAKENHI Grant Numbers JP19H01138, JP17H00731, JP18H03242, JP18H03342, JP19H04154, JP19K02973.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mai, T.; Morihiko, T.; Keiichi, Y. A Monitoring Support System for Elderly Person Living Alone through Activity Sensing in Living Space and Its Evaluation. *IPSJ SIG Notes* **2014**, *2014*, 1–7.
2. Tamamizu, K.; Sakakibara, S.; Saiki, S.; Nakamura, M.; Yasuda, K. Capturing Activities of Daily Living for Elderly at Home based on Environment Change and Speech Dialog. *IEICE Tech. Rep.* **2017**, *116*, 7–12.
3. Alam, M.A.U. Context-aware multi-inhabitant functional and physiological health assessment in smart home environment. In Proceedings of the 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kona, HI, USA, 13–17 March 2017; pp. 99–100.
4. Gochoo, M.; Tan, T.H.; Liu, S.H.; Jean, F.R.; Alnajjar, F.S.; Huang, S.C. Unobtrusive activity recognition of elderly people living alone using anonymous binary sensors and DCNN. *IEEE J. Biomed. Health Inform.* **2018**, *23*, 693–702.
5. Ni, Q.; Garcia Hernando, A.B.; la Cruz, D.; Pau, I. The elderly's independent living in smart homes: A characterization of activities and sensing infrastructure survey to facilitate services development. *Sensors* **2015**, *15*, 11312–11362.
6. Sharmila.; Kumar, D.; Kumar, P.; Ashok, A. Introduction to Multimedia Big Data Computing for IoT. In *Multimedia Big Data Computing for IoT Applications: Concepts, Paradigms and Solutions*; Tanwar, S.; Tyagi, S.; Kumar, N., Eds.; Springer Singapore: Singapore, 2020; pp. 3–36. doi:10.1007/978-981-13-8759-3_1.

7. Singh, A.; Mahapatra, S. Network-Based Applications of Multimedia Big Data Computing in IoT Environment. In *Multimedia Big Data Computing for IoT Applications*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 435–452.
8. Talal, M.; Zaidan, A.; Zaidan, B.; Albahri, A.; Alamoody, A.; Albahri, O.; Alsalem, M.; Lim, C.; Tan, K.L.; Shir, W.; et al. Smart home-based IoT for real-time and secure remote health monitoring of triage and priority system using body sensors: Multi-driven systematic review. *J. Med. Syst.* **2019**, *43*, 42.
9. Microsoft Azure. Object detection - Computer Vision - Azure Cognitive Services | Microsoft Docs. Available online: <https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/concept-object-detection> (accessed on 9 January 2020).
10. IBM Cloud. Speech to Text - IBM Cloud API Docs. Available online: <https://cloud.ibm.com/apidocs/speech-to-text/speech-to-text> (accessed on 9 January 2020).
11. Google Cloud. Cloud Natural Language API documentation. Available online: <https://cloud.google.com/natural-language/docs/> (accessed on 9 January 2020).
12. Triemvitaya, N.; Butsri, S.; Temtanapat, Y.; Suksudaj, S. Sound Tooth: Mobile Oral Health Exam Recording Using Individual Voice Recognition. In Proceedings of the 2019 4th International Conference on Information Technology (InCIT), Bangkok, Thailand, 24–25 October 2019; pp. 243–248.
13. Alexakis, G.; Panagiotakis, S.; Fragkakis, A.; Markakis, E.; Vassilakis, K. Control of Smart Home Operations Using Natural Language Processing, Voice Recognition and IoT Technologies in a Multi-Tier Architecture. *Designs* **2019**, *3*, 32.
14. Lee, H.T.; Chen, R.C.; Chung, W.H. Combining Voice and Image Recognition for Smart Home Security System. In *International Conference on Frontier Computing*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 212–221.
15. Microsoft. Computer Vision | Microsoft Azure. Available online: <https://azure.microsoft.com/en-us/services/cognitive-services/face/> (accessed on 26 December 2019).
16. IBM. Watson Visual Recognition. Available online: <https://www.ibm.com/watson/services/visual-recognition/> (accessed on 23 July 2018).
17. Chen, S.; Saiki, S.; Nakamura, M. Integrating Multiple Models Using Image-as-Documents Approach for Recognizing Fine-Grained Home Contexts. *Sensors* **2020**, *20*, 666. doi:10.3390/s20030666.
18. Chen, S.; Saiki, S.; Nakamura, M. Towards Affordable and Practical Home Context Recognition: - Framework and Implementation with Image-based Cognitive API-. *Int. J. Netw. Distrib. Comput. (IJNDC)* **2019**, *8*, 16–24. doi:10.2991/ijn/dc.k.191118.001.
19. Chen, S.; Saiki, S.; Nakamura, M. Evaluating Feasibility of Image-Based Cognitive APIs for Home Context Sensing. In Proceedings of the International Conference on Signal Processing and Information Security (ICSPIS 2018), Dubai, UAE, 7–8 November 2018; pp. 5–8.
20. Kamishima, T. Clustering. Available online: <http://www.kamishima.net/archive/clustering.pdf> (accessed on 23 July 2018).
21. Muflikhah, L.; Baharudin, B. Document clustering using concept space and cosine similarity measurement. In Proceedings of the 2009 International Conference on Computer Technology and Development, Kota Kinabalu, Malaysia, 13–15 November 2009; Volume 1, pp. 58–62.
22. Lee, L.H.; Wan, C.H.; Rajkumar, R.; Isa, D. An enhanced Support Vector Machine classification framework by using Euclidean distance function for text document categorization. *Appl. Intell.* **2012**, *37*, 80–99.
23. Cormack, R.M. A review of classification. *J. R. Stat. Soc. Ser. A Gen.* **1971**, *134*, 321–353.
24. Milligan, G.W. An examination of the effect of six types of error perturbation on fifteen clustering algorithms. *Psychometrika* **1980**, *45*, 325–342.
25. Milligan, G.W. A Monte Carlo study of thirty internal criterion measures for cluster analysis. *Psychometrika* **1981**, *46*, 187–199.
26. Milligan, G.W. An algorithm for generating artificial test clusters. *Psychometrika* **1985**, *50*, 123–127.
27. Wagstaff, K.; Cardie, C.; Rogers, S.; Schrödl, S. Constrained k-means clustering with background knowledge. In Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williamstown, MA, USA, 28 June–1 July 2001; Volume 1, pp. 577–584.
28. Cheng, Y. Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* **1995**, *17*, 790–799.

29. Tax, N.; Sidorova, N.; Haakma, R.; van der Aalst, W.M. Mining local process models with constraints efficiently: applications to the analysis of smart home data. In Proceedings of the 2018 14th International Conference on Intelligent Environments (IE), Rome, Italy, 25–28 June 2018; pp. 56–63.
30. Hu, S.; Tang, C.; Liu, F.; Wang, X. A distributed and efficient system architecture for smart home. *Int. J. Sens. Netw.* **2016**, *20*, 119–130.
31. Xu, K.; Wang, X.; Wei, W.; Song, H.; Mao, B. Toward software defined smart home. *IEEE Commun. Mag.* **2016**, *54*, 116–122.
32. Copos, B.; Levitt, K.; Bishop, M.; Rowe, J. Is anybody home? Inferring activity from smart home network traffic. In Proceedings of the 2016 IEEE Security and Privacy Workshops (SPW), San Jose, CA, USA, 22–26 May 2016; pp. 245–251.
33. Stojkoska, B.R.; Trivodaliev, K. Enabling internet of things for smart homes through fog computing. In Proceedings of the 2017 25th Telecommunication Forum (TELFOR), Belgrade, Serbia, 21–22 November 2017; pp. 1–4.
34. Roelleke, T.; Wang, J. TF-IDF Uncovered: A Study of Theories and Probabilities. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*; ACM: New York, NY, USA, 2008; SIGIR '08, pp. 435–442. doi:10.1145/1390334.1390409.
35. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at International Conference on Learning Representations (ICLR 2013), May 2013, Scottsdale, Arizona.
36. Le, Q.; Mikolov, T. Distributed Representations of Sentences and Documents. In Proceedings of the 31st International Conference on International Conference on Machine Learning—Volume 32, Beijing, China, 22–24 June 2014; ICML'14, pp. II–1188–II–1196.
37. Pennington, J.; Socher, R.; Manning, C.D. GloVe: Global Vectors for Word Representation. Available online: <https://nlp.stanford.edu/projects/glove/> (accessed on 15 April 2019).
38. Research, F. fastText: A library for efficient learning of word representations and sentence classification. Available online: <https://github.com/facebookresearch/fastText> (accessed on 15 April 2019).
39. Ye, J. Cosine similarity measures for intuitionistic fuzzy sets and their applications. *Math. Comput. Model.* **2011**, *53*, 91–97.
40. Yen, L.; Vanvyve, D.; Wouters, F.; Fouss, F.; Verleysen, M.; Saerens, M. Clustering using a random walk based distance measure. In Proceedings of the ESANN, Bruges, Belgium, 27–29 April 2005; pp. 317–324.
41. Huang, A. Similarity measures for text document clustering. In Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008), Christchurch, New Zealand, 14–18 April 2008; Volume 4, pp. 9–56.
42. Van der Laan, M.; Pollard, K.; Bryan, J. A new partitioning around medoids algorithm. *J. Stat. Comput. Simul.* **2003**, *73*, 575–584.
43. Sheikholeslami, G.; Chatterjee, S.; Zhang, A. Wavecluster: A multi-resolution clustering approach for very large spatial databases. In Proceedings of the VLDB, New York, NY, USA, 24–27 August 1998; Volume 98, pp. 428–439.
44. Noble, W.S. What is a support vector machine? *Nat. Biotechnol.* **2006**, *24*, 1565–1567.
45. Baum, E.B.; Wilczek, F. Supervised Learning of Probability Distributions by Neural Networks. *Neural Information Processing Systems*; Anderson, D.Z., Ed. American Institute of Physics, 1988, pp. 52–61.
46. Geurts, P.; Irrthum, A.; Wehenkel, L. Supervised learning with decision tree-based methods in computational and systems biology. *Mol. Biosyst.* **2009**, *5*, 1593–1605.
47. Microsoft. Computer Vision | Microsoft Azure. Available online: <https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/> (accessed on 27 November 2019).
48. Clarifai. Enterprise AI Powered Computer Vision Solutions | Clarifai. Available online: <https://clarifai.com/> (accessed on 15 April 2019).
49. Imagga. Imagga API. Available online: <https://docs.imagga.com/> (accessed on 15 April 2019).
50. ParallelDots. Image Recognition. Available online: <https://www.paralldots.com/object-recognizer> (accessed on 15 April 2019).

51. Arthur, D.; Vassilvitskii, S. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.
52. Azure Machine Learning Studio. Available online: <https://azure.microsoft.com/ja-jp/services/machine-learning-studio/> (accessed on 1 February 2019).



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).