 doi:10.20944/preprints202108.0163.v1

# WikiServe: Using Wikipedia to Match IoT based Services for Situation Response

**Sazid Zaman Khan · Alan Colman · Iqbal H. Sarker**

**Abstract** A large number of smart devices (things) are being deployed with the swift development of Internet of Things (IOT). These devices, owned by different organizations, have a wide variety of services to offer over the web. However, *appropriate service matching methods* are required to find the relevant services. Organizations that manage situation responses and organizations that provide IOT services are likely to be independent of each other, and therefore it is difficult for them to adopt a common ontological model to facilitate the service matching. Moreover, there exists a large conceptual gap between the domain of discourse for situations and the domain of discourse for services, which cannot be adequately bridged by existing techniques. In this paper, we address these issues and propose a new method, *WikiServe*, to identify IOT services that are functionally relevant to a given situation. Using concepts (terms) from situation and service descriptions, WikiServe employs Wikipedia as a knowledge source to bridge the conceptual gap between situation and service descriptions and match functionally relevant IOT services for a situation. WikiServe performs better than a commonly used baseline method in terms of Precision, Recall and F measure for service matching.

**Keywords** Semantics · Internet of Things · Situation Response · Services in Internet of Things

Sazid Zaman Khan
International Islamic University Chittagong, E-mail: szkhanctg@gmail.com

Alan Colman
Swinburne University of Technology, VIC-3132, Australia.

Iqbal H. Sarker
Dept. of Computer Science and Engineering, Chittagong University of Engineering & Technology, E-mail: iqbal@cuet.ac.bd
Correspondence: szkhanctg@gmail.com, iqbal@cuet.ac.bd

## 1 Introduction

Organizations are equipping themselves with a wide variety of smart things and exposing them as services over the web. In the near future, owners of IOT infrastructures may offer services from their smart things, which will be vital during emergent (unplanned) situations such as traffic accident, bushfire, etc. However, not all services would be relevant to all situations. Hence, there is a need to have automated methods that can match potentially relevant services to help respond to a given situation, by considering their functional and contextual relevance. In this paper, we focus on only the *functional relevance* of the services to a given situation. As the real world application dictates, our research challenge involves service matching without explicit requirements being specified by the situation responders. Instead, service matching needs to rely only on a high-level description of the situation (as provided by the human responders) as well as the available IOT service descriptions. If a situation occurs in an IOT instrumented environment, it would be highly advantageous for the situation responders to have a list of potentially useful and relevant IOT services among the hundreds and even thousands of available IOT services.

Due to the conceptual gap between the domains of discourse for services and situations, relating these two domains is rather indirect and far more difficult than relating service offerings to service requirements, where there has been some limited existing research effort.

Ontology based IOT service matching are widely studied in existing literature [16,18]. These works often refer to and maintain compatibility with the Semantic Sensor Network (SSN) ontology [4]. Ontological approaches work best within a single domain of discourse. Our problem involves two very different domains of dis-

 doi:10.20944/preprints202108.0163.v1



course (situations and IOT services). Given that the responsible organizations are often independent of each other, it would be extremely difficult if not impossible for them to agree on and maintain a common ontology, achieving IOT service matching across these different domains of discourse.

In contrast, Natural Language Processing (NLP) based service matching methods [14] do not impose detailed concept structures for service matching. Instead, they derive conceptual vectors from natural language documents or use text corpus. Often these techniques extract latent topics from service descriptions and relate them to service queries. However, the conceptual gap between the situation domain and the IOT service domain is much larger than that between specific service requirements and service offerings, limiting the usefulness of the existing methods for service matching. Other web knowledge based approaches for service/API matching either use crowd knowledge to bridge queries (requirements) and services, or expand keywords for matching services. These methods still focus on requirements based service matching, and their suitability to situation-oriented service matching is unclear.

We utilize the concepts (terms) from situation descriptions and IOT based service descriptions to find services that are functionally relevant to a situation. We propose a method, named *Wiki-Serve*, which uses Wikipedia knowledge source to bridge the conceptual gap between the terms from the situation and service descriptions and to match functionally relevant services to the given situation. In particular, we make the following contributions:

- We demonstrate that Wikipedia can be used as an organized and reliable knowledge source to relate situations and IOT services.
- Our method for service matching uses weighted situation related articles from Wikipedia and weighted occurrences of IOT services terms in these articles to calculate the relevance scores of services for a particular situation, consequently creating a ranked list of services for the situation.

Our experimental results show that the proposed method can indeed perform satisfactorily close to human matching of relevant IOT services for a situation and performs better than the Word2Vec based relevance matching. Rest of the paper is organized as following: Section 2 describes a motivating scenario and analyzes the requirements. Section 3 reviews the related work. Section 4 describes our method, while Section 5 presents the experimental results. Section 6 concludes the paper.

## 2 Motivating Scenario and Requirements

Let us consider an IOT instrumented scenario for the "Great Ocean Road" which is a popular tourist spot in Victoria. Various sensors/actuators exist in the area and are owned by different organizations. Sensors from the Bureau of Meteorology (BOM) detect smoke, temperature, wind speed and directions. The State Highway Authority (SHA) has its own internet-connected devices installed, such as visibility meters, air quality sensors and smart road signs.

A hot summer day triggers a bushfire near the highway. The Victoria State Emergency Service (SES) is the responsible management authority for such an emergency with the various operational manuals and procedures, and would like to use the services from the smart things in the area as part of its response to the bushfire. For example, the air quality sensor service can be used to measure the air pollution. However, not all thing-based services will be relevant to the situation. For example, the service from a tsunami detection buoy is not relevant for a bushfire. It would be highly desirable to have automated methods that can identify and match the functionally relevant IOT services among the hundreds and even thousands of services offered by different organizations for such a bushfire situation.

The above motivating scenario highlights the constraints and requirements for automated situation-oriented service matching:

1. The situation response coordination organization like the Victoria SES and the service provider organizations are usually disparate and independent of each other.

2. Descriptions of the situation and services may be the only direct information available for identifying the relevant and potentially useful IOT services for the situation.

3. The conceptual gap between the realm of the situation and the realm of services is huge. For example, the description of a bushfire is about the relevant state of the physical world, while the service descriptions concern their functionality and operation. To be able to bridge this large conceptual gap and automatically identify the relevant services to the situation is the challenge.

## 3 Related Work

In this section, we review the related works on functionality-oriented NLP-based matching, selection and recommendation of services, including IOT services.

Lin et al. [14] aim to recommend services using only natural language service descriptions. Their work notes

the deficiency of Latent Dirichlet Allocation (LDA) [6] for short text documents. In their first approach, they use Topic Modeling with Non-negative Matrix Factorization on Term Correlation matrix (TNMF) [25] to extract topics and associate topics to services. In two other approaches, they use Topic Modeling with TNMF and Louvain's Community Detection [8], Long Short Term Memory based auto encoder [23] [21] and K-means clustering [22] to extract topics and communities. After associating services to communities, they match services by computing similarity scores between a query and the (service) topics using the Wu-Palmer score [24] and consequently recommend services based on the similarity scores.

Liu et al. [15] improve Latent Dirichlet Allocation (LDA) extracted topics from short text descriptions of service requests and service descriptions (both are raw text). Their method requires human involvement in topic recognition and uses the Bayesian model [22] and topic/word vectors for matchmaking between services and requests. Their method provides significant improvement in Recall and Precision for service matching over LDA and Probabilistic Latent Semantic Analysis [12].

Cassar et al. [7] aim to combine probabilistic service matchmaking with logical service signature matching to match services and their input/output parameters with those of a request. LDA [22] is used to extract latent factors from service descriptions and queries. Then they represent the services and query as vectors of these latent factors. They use Cosine similarity measure to calculate the similarity between the services and the query vector and consequently select the relevant services.

Hao et al. [11] aim to bridge the language gap between mashup developers and service providers. They note that application scenarios are often not clear in the service query descriptions of the developers. Therefore, they propose a technique, named Targeted Reconstruction of Service Description. Their technique uses LDA to extract topics from mashup descriptions and queries. Then they use Jensen-Shannon divergence to find the similarity between the mashup descriptions and the queries. While reconstructing descriptions, they use service usage history in application scenarios. This is only suitable when a history of service invocations is available.

Mohammad et al. [19] utilize questions and accepted answers of the stackoverflow.com website to match keywords from queries to APIs with the aim to improve service matching. However, the API descriptions and query keywords are still expressed at similar levels of abstraction. Guinard et al. [10] propose query augmentation to achieve better service type lookup. It uses search engines and Wikipedia to expand the initial service type keywords such as "smart meter". The expanded keywords are then used for searching services. While such query expansion could be a good strategy to append additional keywords for improved service search, the provided and expanded keywords are again very much in domains of similar abstraction.

In all of the approaches discussed above, services are matched, selected or recommended based on well specified requirements or queries, with or without keyword/knowledge expansion. The requirements/queries for services are expressed at a level of abstraction that is very close to the service descriptions. Therefore, the chance of convergence at the latent factor space between the requirements and service descriptions is high. For our problem, the applicability of these approaches is limited because the large conceptual gap between the domains of discourses for situations and services makes convergence between them in the latent factor space difficult.

## 4 Methodology

### 4.1 Overview

The main obstacle in matching functionally relevant services to a situation is the large conceptual gap between the service domain and the situation domain. This gap must be bridged by some knowledge sources that can relate situation and service concepts (terms). Google search APIs (linking to millions of websites) may be the largest knowledge source. However, these distinct web sites and web sources cannot be considered a single organized source of knowledge. For example, consider the search results for "Bushfire". Among the top search results, some are websites that only contain maps of bushfire incidents with little text, while others are news articles of varying lengths. The heterogeneity of the types of web resources makes them extremely difficult to use for systematically relating situations and services.

In general, more organized web knowledge sources are required that have lesser level of heterogeneity. According to this point, Wikipedia is probably the best web based knowledge source [9]. In [9], authors created vectors for text documents using Wikipedia and then used vector similarity measures to find relatedness of text documents. Our work also uses Wikipedia as the knowledge source. However, our purpose is to close the conceptual gap between the situation and service domains, more specifically relating the specific concepts or terms from these domains. We assume that the occurrences of a service's terms in a Wikipedia article about

 doi:10.20944/preprints202108.0163.v1



a situation or its term should indicate the relevance of the service to the situation. To calculate the specific relevance scores of services to a situation, WikiServe carries out a series of processing on the situation terms, service terms and Wiki articles about these terms.

Fig. 1 provides a high-level overview of the WikiServe method. As generally is the case in practice, we assume that descriptions for situations and IOT services are available that provide the categories, capabilities and properties of situations/services. Key situation terms such as situation category and properties can be obtained from the situation description using feature extraction techniques like those found in [13]. Similarly, key service terms such as service category and service capabilities can be obtained from the service description. Wikipedia articles with situation terms as titles are obtained that serve as the knowledge sources for the situation. These articles are weighted to account for their significance, the main situation article (e.g. Bushfire) being most significant source of knowledge, receives the highest weight.

For service terms, Wikipedia articles with the terms as titles are obtained and these articles are used to weight service terms. The service terms are weighted so that service capability terms receive highest weight assuming that they are more important than service category terms. WordNet synonyms are obtained for service terms, this makes sure that not only original terms but also their synonyms can be used in service scoring. The weighted situation articles and service terms are then fed to the service scoring process and relevance scores of services are calculated. Finally, a ranked list of services is prepared for a situation based on their relevance scores.

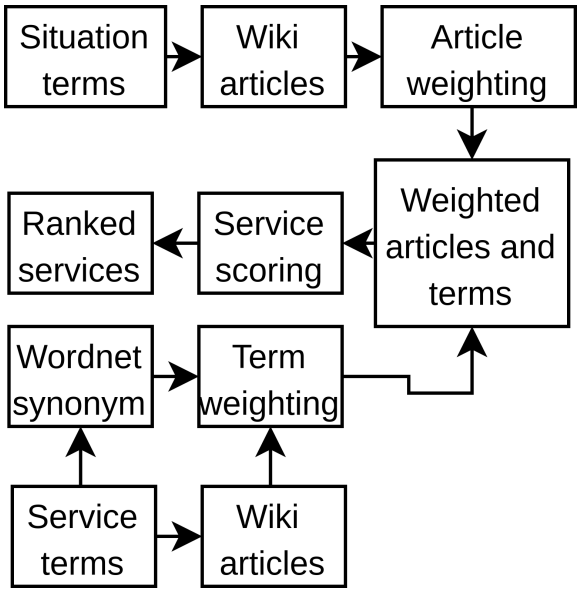


**Fig. 1** Overview of the WikiServe method

## 4.2 Situations, IOT Services and Their Key Terms

We use three key aspects of a situation and obtain the corresponding terms from the situation description. These are the situation category, the situation properties and the associated situation category. The situation category expresses the type of the situation, for example, whether it is a Bushfire or a Flood. The associated situation category expresses the type of the associated situation if there is an associated situation. For example, a Bushfire can have an associated situation, a Drought. Situation properties express core and contextual properties of a situation. For example, to describe a Bushfire situation, the situation coordinators may use fire intensity and maximum flame height as core properties. Furthermore, they may use humidity and temperature as contextual properties. WikiServe uses the key terms from the situation category, associated situation category and situation properties to obtain Wikipedia resources for the situation. In the example situation description used in this paper, the key terms are “Bushfire”, “Drought”, “Fire”, “Flame”, “Humidity” and “Temperature”. These terms are obtained from the situation description.

Since the service capabilities and service category express the functionality of an IOT service, these two aspects of the service are obtained from the service descriptions and used in service matching. There can be more than one capability for a service; however a service can have only one category. For example, a wind speed and direction sensor service can have two capabilities (“wind speed” and “wind direction”) and a category (“wind”). These key terms (“wind speed”, “wind direction” and “wind”) are looked up in Wiki articles for the situation by WikiServe during the service scoring process.

## 4.3 Situation Terms Processing

The Wiki articles for situation terms serve as the knowledge sources for the situation. Therefore, WikiServe uses the key terms from situation category, associated situation category and situation properties to obtain situation articles from Wikipedia. For the Bushfire example, these terms are: Bushfire, Drought, Fire, Flame, Humidity and Temperature. We retrieve the Wikipedia articles that have these terms as their titles. The Wikipedia REST API [1] is used to download Wikipedia articles for the situation. Hyperlinks, markups and stop words

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 August 2021 doi:10.20944/preprints202108.0163.v1



such as "a", "the", "this", "here", "there", etc are removed because these words do not carry any significant meaning. Then the Porter stemming algorithm [2] is used to get the root terms of the words present in the Wikipedia article. Stemming is done so that different inflected forms of a word such as "winds" are transformed to the root word "wind".

The situation is "Bushfire". "Fire", "Flame", "Humidity" and "Temperature" are terms from properties of the situation while "Drought" is the associated situation category. Since the situation is Bushfire, the main situation article (i.e. the main concept) is Bushfire. The "Bushfire" article is the main knowledge source and the articles with other titles (e.g."Fire", "Drought") are secondary knowledge sources. Therefore, a service should receive higher score for its terms appearing in the Bushfire article than their appearing in other situation term articles. Thus, we discriminate among service term occurrences in different articles by weighting the Wikipedia articles of a situation. To achieve this, we assume that the article with the situation category term as the title has the highest weight, 1. For example, the "Bushfire" Wikipedia article has the highest weight, 1, as Bushfire is the situation category. We consider the weights for other situation articles (e.g. Drought, Fire, Flame, etc) as being the values of their WikiCosineSimilarity (this similarity measure is explained in sub section 4.5) measure with the Bushfire article. The situation article's weight is labeled $\mathrm{WSIT_i}$.

Algorithm-1 shows the procedure for calculating the weight of the articles. In the algorithm, the first line assigns a weight of 1 to the main situation article. Lines 3 to 4 assign weights to other situation articles (e.g. Drought, Fire) based on their WikiCosineSimilarity (WikiCSim) with the main situation article.

**Algorithm 1:** Situation article weight calculation

**Input :** Situation category article C and other N situation articles $[sit_1, \cdots sit_N]$
**Output:** N+1 weighted situation articles

1 $W_C \leftarrow 1$
2 **for** $i \leftarrow 1$ **to** $N$ **do**
3 | Calculate $WikiCSim_i$ between C and $sit_i$
4 | $WSIT_i \leftarrow WikiCSim_i$
5 **end for**
6 **return** *Weighted situation articles*

## 4.4 Service Terms Processing

Service capability terms are clear expressions of the IOT service functionality. For a wind speed and wind direction sensor service, the capability terms "wind speed" and "wind direction" unambiguously convey their functionality.

In contrast, service category term "wind" for the service only conveys a rough sense of what the service is about. Therefore, service capability terms are given the highest weight (a weight of 1). The weights for the service category terms are calculated based on their WikiCosineSimilarity with the service capability terms, being less than 1. Consequently, a service receives higher score for the occurrence of its capability terms than the occurrence of its category term in situation articles. Algorithm-2 shows the procedure for calculating weights of service terms. In the algorithm, line 2 assigns a weight of 1 to the service capability terms. Rest of the algorithm assigns weights to service category term based on it's maximum WikiCosineSimilarity with service capability terms.

**Algorithm 2:** Service term weight calculation

**Input :** Service category term T and other K service capability terms $[tcap_1, \cdots tcap_K]$
**Output:** K+1 weighted service terms

1 **for** $i \leftarrow 1$ **to** $K$ **do**
2 | $WTCAP_i \leftarrow 1$
3 **end for**
4 $max \leftarrow 0$
5 **for** $i \leftarrow 1$ **to** $K$ **do**
6 | Calculate $WikiCSim_i$ between T and $tcap_i$
7 | **if** $WikiCSim_i > max$ **then**
8 | | $max \leftarrow WikiCSim_i$
9 | **else**
10 | | **continue**
11 | **end if**
12 **end for**
13 $W_T \leftarrow max$
14 **return** *Weighted service terms*

To lessen the chance of missing synonyms of service terms in situation articles, we use WordNet [5] to find the synonyms of the service terms. The first synset of WordNet provides the most accurate synonyms for a term, hence we only use the synonyms from this synset. Our experiments have shown that synonyms of multiword terms result in the drift of the meaning of the original term. Hence, WordNet based expansion of terms is done only if the service term is a single word. As an example, for the wind speed and direction sensor service, we retrieve the first synset synonyms "air current" and "current of air" for the term "wind". These two synonyms are given the same weight as "wind".

Similar to situation articles, service terms are stemmed using the Porter stemmer. Doing the same stemming process for service terms means that if the situation ar-

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 August 2021 doi:10.20944/preprints202108.0163.v1



ticle originally had "winds" and the service term has "wind", there is still a match between the two.

### 4.5 WikiCosineSimilarity Calculation

Algorithm-3 shows the WikiCosineSimilarity calculation. WikiCosineSimilarity is a Wikipedia based Cosine similarity measure. On the situation side, it is used to weight other situation articles relative to the main article. On the IOT service side, it is used to weight service category term relative to the service capability terms. For example, on the situation side, given the main situation article Bushfire and another situation article Drought, two 50 dimensional vectors for both articles are derived by using the top 50 most frequent words appearing in each article. Then, the Cosine similarity of these two vectors is calculated and this similarity is assigned as the weight of the Drought article which will be a value less than 1. Similarly, weights of other/secondary situation articles are calculated.

In our Bushfire example situation, the Fire article had the highest weight among the secondary situation articles, a value of 0.425. On the service side, the Wikipedia articles with IOT service capability terms and service category term as titles are obtained and vectorized as described above. The articles with service capability terms as titles are considered the main articles. For example, the articles "wind speed" and "wind direction" are obtained and vectorized since "wind speed" and "wind direction" are capability terms of the wind speed and wind direction sensor service. Then the Wiki article "wind" for the category term "wind" is obtained and vectorized. Later, the Cosine similarity of "wind" article vector with both the "wind speed" and "wind direction" article vectors are calculated and the higher weight is assigned as the weight of the term "wind".

In the algorithm, line 1 creates the vector for the articles using top 50 frequent words of the articles. Lines 2 and 3 calculate and assign WikiCosineSimilarity using the vectorized articles.

**Algorithm 3:** WikiCosineSimilarity Calculation

**Input :** Main Wiki article $W_m$, candidate Wiki article $W_d$
**Output:** WikiCosineSimilarity between $W_m$ and $W_d$

1 Calculate vector $V_m$ and $V_d$ using top 50 frequent words of $W_m$ and $W_d$ respectively
2 Calculate Cosine Similarity $CosineSim_{vmvd}$ between $V_m$ and $V_d$
3 $WikiCSim \leftarrow CosineSim_{vmvd}$
4 **return** *WikiCosineSimilarity*

### 4.6 Service Score Calculation

To quantify the relevance of IOT services to a situation, the scores of the services are calculated. The aim is to rank the services based on their score for a situation. A service gets a score for each of its term occurring in situation articles. The score is calculated by multiplying the number of occurrence(s) of the term in an article by the term weight and the article weight. Then the total score of a service for a situation is calculated by summing the scores of all of the service's terms across all the situation articles. Eqn.1 is used to calculate the total score of a service.

$$S_{sv} = \sum_{j=1}^{k} \sum_{i=1}^{n} Occ_{ij} * Wsvt_i * Wsita_j \quad (1)$$

where $Occ_{ij}$ is the number of occurrences of the service term i in the situation article j, $Wsvt_i$ is the weight of the service term i, $Wsita_j$ is the weight of the situation article j, n is the total number service terms of a service, k is total number of situation articles for a situation.

Algorithm-4 shows the procedure for calculating the total score of services and creating the ranked list of matched services.

**Algorithm 4:** Service score and ranked list calculation

**Input :** Situation articles $[sit_1, \cdots sit_p]$ for a situation, services $[serv_1, \cdots serv_q]$, service terms for each service
**Output:** Ranked list of services according to service scores

1 **for** $i \leftarrow 1$ **to** $q$ **do**
2  $Score_{servi} \leftarrow 0$
3  **for** $j \leftarrow 1$ **to** $p$ **do**
4  **foreach** *service term* $t \in serv_i$ **do**
5  $Score_{servi} \leftarrow Score_{servi} + Occ_{tsitj} * W_t * W_{sitj}$
6  **end foreach**
7  **end for**
8  Add $Score_{servi}$ to ServiceScoreList
9 **end for**
10 Sort ServiceScoreList by descending order
11 **return** *ServiceScoreList*

In the algorithm, Lines 1 to 9 calculate score of a service based on its term occurrence, term weight and situation article weight. Line 10 and 11 calculate and return the ranked list of services for the situation. For the Bushfire situation example, the Fire sprinkler service get the top position of the ranked list with scores 199.316.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 August 2021 doi:10.20944/preprints202108.0163.v1



## 5 Experiment and Results

### 5.1 Experiment Setup

We have experimented with five natural disaster situations: Bushfire, Cyclone, Flood, Landslide and Earthquake. We use the terms from categories, properties and associated situations for these situations in our WikiServe method. We consider environments with IOT services installed along highways, national parks, forests, coastlines, as described in Section 2. We have gathered over 100 such services as service samples for our experiment. Most of these services are based on devices listed in [3], while others are taken from websites of respective manufacturers of the devices. Some examples of these services are the firefighting drone service, water sprinkler service, weather station service, and rainfall sensor service. According to human judgment, some of these services are deemed to be relevant for some of our situations while others are not. Five ideal reference lists of services (one for each situation) are created by a human judge. The lists are prepared by choosing services from the service samples. The lists of services are ranked based on the perceived relevance of the services to the situations. We compare the Precision, Recall and F-measure of the WikiServe method and a method that is based on Word2Vec [17] while keeping the human prepared list as the (ideal) baseline. For the Word2Vec based method, we use the "Average Feature Vector" of situation terms and service terms. In the Word2Vec based method, we create two average feature vectors using situation terms (e.g. Bushfire, Fire, Flame, Humidity, and Temperature) and service terms (e.g. wind speed, wind direction, wind). Then the similarity score of these two feature vectors is calculated. The average feature vector of every sample service is compared with the average feature vector of the situation and this gives us a score for each service. The score is between 0 and 1, a high score implies a high correlation between the service and the situation. Using these scores, a ranked list of services for a situation is created.

Precision is the ratio of correctly matched IOT services to the sum of correctly and incorrectly matched services in the ranked lists created by a method (i.e., WikiServe based method). Recall is the ratio of correctly matched services to the sum of correctly matched and missed services in the ranked lists created by a method (i.e., Word2Vec based method). A service is considered correctly matched if it also exists in the ideal baseline list of IOT services for a situation prepared by the human judge. A service is considered incorrectly matched if it does not exist in the ideal baseline list of IOT services for a situation prepared by the human judge but exists in the ranked list created by a method (i.e., WikiServe or the Word2Vec based method). A service is considered missed if it exists in the ideal baseline list of IOT services for a situation prepared by the human judge but does not exist in the service ranked list created by a method (i.e., WikiServe or theWord2Vec based method). The formula for Precision, Recall and F measure are given in Eqn. (2), (3) and (4) respectively.

$$P = svt_c/(svt_c + svt_ic) \tag{2}$$

$$R = svt_c/(svt_c + svt_m) \tag{3}$$

$$F_1 = 2 * P * R/(P + R) \tag{4}$$

where $\text{svt}_\text{c}$ is the total number of correctly matched services, $\text{svt}_\text{ic}$ is the total number of incorrectly matched services, $\text{svt}_\text{m}$ is the total number of missed services. P stands for precision and R stands for recall.

### 5.2 Experiment Results

In Fig.2 and Fig.3, we compare the Precision and Recall of the WikiServe and the Word2Vec based method for the 5 situations.

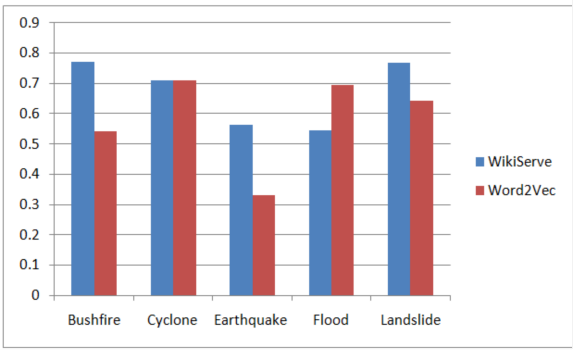


**Fig. 2** Comparison of precision of WikiServe and Word2Vec based method

Fig.2 demonstrates that for 3 out of 5 situations, the precisions of WikiServe are better than those of the Word2Vec based method. For one situation (Cyclone) the Precision of both the methods are same. The average Precision of WikiServe across the 5 situations is 0.672 which is better than the average Precision (0.585) of the Word2Vec based method.

Fig.3 shows the recalls of WikiServe and the Word2Vec based method for the 5 situations. Recalls of WikiServe are better than those of the Word2Vec based method for 4 out of 5 situations. Only for the Flood situation the Recall is worse. Average Recall of WikiServe for the 5 situations is 0.643 which is better than the average Recall (0.570) of the Word2Vec based method.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 August 2021 doi:10.20944/preprints202108.0163.v1



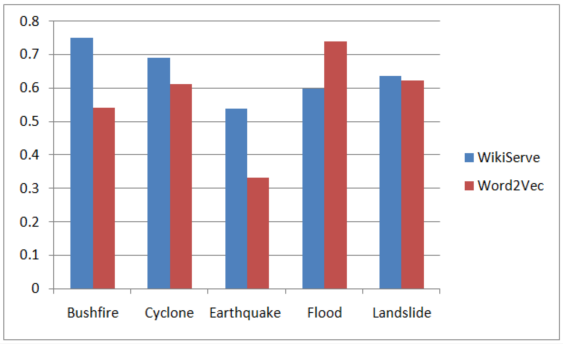


**Fig. 3** Comparison of recall of WikiServe and Word2Vec based method

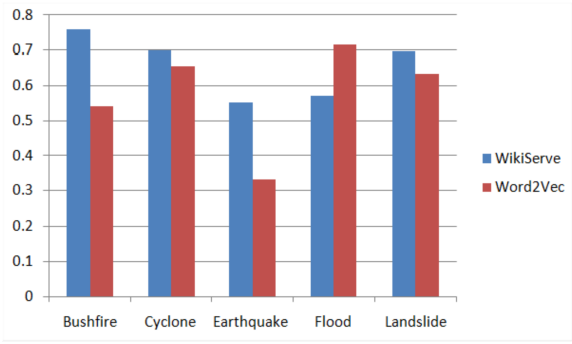


**Fig. 4** Comparison of F-measure of WikiServe and Word2Vec based method

The Word2Vec based method performs relatively well for Cyclone and Flood because there is a good number of water related sensor services in the sample service set. Since Word2Vec uses a big corpus such as Google news dataset, it can estimate the relation between key terms of water related sensor services and the Flood situation. However, the disadvantage of using a (pre trained) big corpus for service relevance estimation is obvious, because more often than not, irrelevant services are estimated to be relevant as demonstrated by the Word2Vec based method's worst performance for majority of situations.

The F-measures for the 5 situations are shown in Fig. 4. The F-measures of WikiServe are better than those of the Word2Vec based method for all situations except Flood.

The average Precision, Recall and F-measure of WikiServe across 5 situations are 8.7, 7.3 and 7.9 percents better than those of the Word2Vec based method respectively. This is due to the fact that, the pre trained Word2Vec model estimates relation between terms using huge corpora (such as Google news dataset), hence high correlation in data [20], particularly, between obviously distant service terms and situation terms are found which do not reflect the reality in terms of their real relationship. However,our approach uses a few relevant Wiki articles for the situation and limits the context of service relevance matching to these articles only instead of a huge corpus.

Relevance is assumed if only the service terms are directly found in the selected few situation term articles. Therefore, WikiServe is more focused and provides more precise results. In Tables 1, 2 and 3, we provide some sample situation article weights, service term weights and service scores that are sample outputs of Algorithm-1, Algorithm-2 and Algorithm-4 respectively.

We consider the results satisfactory for the following reasons:

**Table 1** Sample article weights for different situations

| Situation Name | Term1 and Weight | Term2 and Weight |
|---|---|---|
| Bushfire | Bushfire-1.0 | Drought-0.075 |
| Cyclone | Cyclone-1.0 | Wind-0.27 |
| Flood | Flood-1.0 | Wind-0.214 |

**Table 2** Sample service term weights

| Service Name | Term1 and Weight | Term2 and Weight |
|---|---|---|
| Tsunami buoy | Tsunami-1.0 | Earthquake-0.312 |
| Vibration sensor | Vibration-1.0 | Earthquake-1.0 |
| Rainfall sensor | Rainfall-1.0 | Rain-1.0 |

**Table 3** Sample service scores for different situations

| Situation name | Fire sprinkler | Smoke sensor |
|---|---|---|
| Bushfire | 199.316 | 49.027 |
| Cyclone | 7.011 | 21.30 |
| Earthquake | 4.746 | 3.093 |

- Our aim was to identify "possibly useful" services for a situation rather than precise services for specific requirements. In other words, situation coordinators have not specified any requirements for particular services yet services are matched with relatively good coherence compared to human judgment.
- We are solely depending on Wiki based matching between situations and services. There is no prior agreement or prior ontological link established between the situation and IOT services. From the results, it is evident that Wikipedia can successfully relate IOT services to situations and has performed satisfactorily close to human reasoning. To the best of our knowledge, such an approach for finding thing based services relevant to situations has not been explored previously.
- The experiments have showed that our method can outperform a state of the art relevance matching

mechanism (i.e. Word2Vec) in terms of finding the relevant services for a situation.

## 6 Conclusion and Future Work

In this paper, we have investigated the problem of finding thing/IOT based services the functionalities of which are relevant to a given situation. Given the large number of IOT enabled services that are becoming available, we have introduced the WikiServe method that can automate the task of matching functionally relevant services to a situation. Our method utilizes concepts or terms from the situation and service descriptions. It uses Wikipedia as the knowledge source to bridge the conceptual gap between the service and situation (terms) to find functionally relevant services to a situation. The experimental evaluation has demonstrated that our method is a suitable candidate to automate the task of matching relevant services for emergent situations. It has performed better than Word2Vec based relevance matching and its results are close to human reasoning.

As future work, we plan to explore the incorporation of non-functional contexts (e.g. the location of situations and services) in IOT service matching. Furthermore, we also plan to explore service search methods for specific requirements in given situations (e.g. resident rescue during Bushfire).

**Acknowledgements** We are thankful to Prof. Dr. Jun Han, Software Engineering, Swinburne University of Technology for his suggestions which improved the paper.

## Declarations

Funding is not applicable. There is no conflict of interest. Code and data are available.

## References

1. URL https://en.wikipedia.org/api/
2. URL http://snowball.tartarus.org/
3. Australia iot internet of things solutions supplier. URL https://www.IOT-store.com.au/
4. Semantic sensor network ontology. URL https://www.w3.org/TR/vocab-ssn/
5. What is wordnet? URL https://wordnet.princeton.edu/
6. Blei D.M.and Ng, A., Jordan, M.: Latent dirichlet allocation. Journal of machine Learning research **3**(Jan), 993–1022 (2003)
7. Cassar G.and Barnaghi, P.W.W., Moessner, K.: A hybrid semantic matchmaker for iot services. In: 2012 IEEE International Conference on Green Computing and Communications, pp. 210–216. IEEE (2012)
8. Clauset, A., Newman, M., Moore C., .: Finding community structure in very large networks. Physical review E **70**(6), 066111 (2004)
9. Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: IJcAI, vol. 7, pp. 1606–1611 (2007)
10. Guinard, D., Trifa V.and Karnouskos, S., Spiess, P., Savio, D.: Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services. IEEE transactions on Services Computing **3**(3), 223–235 (2010)
11. Hao, Y., Fan, Y., Tan, W., Zhang, J.: Service recommendation based on targeted reconstruction of service descriptions. In: 2017 IEEE International Conference on Web Services (ICWS), pp. 285–292. IEEE (2017)
12. Hofmann, T.: Unsupervised learning by probabilistic latent semantic analysis. Machine learning **42**(1-2), 177–196 (2001)
13. Johann, T., Stanik, C., Maalej, W.: Safe: A simple approach for feature extraction from app descriptions and app reviews. In: 2017 IEEE 25th International Requirements Engineering Conference (RE), pp. 21–30. IEEE (2017)
14. Lin, C., Kalia, A., Xiao, J., Vukovic, M., Anerousis, N.: Nl2api: A framework for bootstrapping service recommendation using natural language queries. In: 2018 IEEE International Conference on Web Services (ICWS), pp. 235–242. IEEE (2018)
15. Liu, Y., Zhu, T., Jiang, Y., Liu, X.: Service matchmaking for internet of things based on probabilistic topic model. Future Generation Computer Systems **94**, 272–281 (2019)
16. Manno G., S.W.S.L., Taccari, G.: A semantic-based federated cloud system for emergency response. Concurrency and Computation: Practice and Experience **27**(13), 3316–3344 (2015)
17. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
18. Perera, C., Zaslavsky, A., Liu, C.H., Compton, M., Christen, P., Georgakopoulos, D.: Sensor search techniques for sensing as a service architecture for the internet of things. IEEE Sensors Journal **14**(2), 406–420 (2013)
19. Rahman, M., Roy, C., Lo, D.: Rack: Automatic api recommendation using crowdsourced knowledge. In: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), vol. 1, pp. 349–359. IEEE (2016)
20. Sarker, I.H.: Data science and analytics: An overview from data-driven smart computing, decision-making and applications perspective. SN Computer Science pp. 1–22 (2021)
21. Sarker, I.H.: Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. SN Computer Science (2021)
22. Sarker, I.H.: Machine learning: Algorithms, real-world applications and research directions. SN Computer Science **2**(3), 1–21 (2021)
23. Schmidhuber, J., Hochreiter, S.: Long short-term memory. Neural Comput **9**(8), 1735–1780 (1997)
24. Wu, Z., Palmer, M.: Verb semantics and lexical selection. arXiv preprint cmp-lg/9406033 (1994)
25. Yan, X., Guo J.and Liu, S.C.X., Wang, Y.: Learning topics in short texts by non-negative matrix factorization on term correlation matrix. In: proceedings of the 2013 SIAM International Conference on Data Mining, pp. 749–757. SIAM (2013)