Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2017) 25: 2784 – 2795

doi:10.3906/elk-1503-61

Matching points of interest with user context: an ANN approach

Özgün YILMAZ*
Department of Computer Engineering, Ege University, İzmir, Turkey

Received: 08.03.2015 • **Accepted/Published Online:** 01.11.2016 • **Final Version:** 30.07.2017

Abstract: In this paper, the design and development of an artificial neural network (ANN) for similarity value calculation in a context-aware system is proposed. This neural network is used by the neural agent of the iConAwa system. Since iConAwa is an intelligent, context-aware, multiagent system, it provides mobile users with context-aware information and services, and also provides communication with each other. Context and points of interest are modeled in a flexible and extensible way by using ontologies. iConAwa derives high-level implicit context from low-level explicit context by inference performed over the context ontology. This approach decouples context reasoning from the source code of the system. With the addition of a neural agent, which uses an ANN, the system has learning capability. By using a neural network for similarity value calculation, the system can adapt to the needs of different people. System owners can introduce their own similarity metric considering their own requirements, which further improves the iConAwa system. Thus, the extended iConAwa system combines expert system characteristics with the capability to learn.

Key words: Context awareness, context ontology, context reasoning, artificial neural network, multiagent system, intelligent system

1. Introduction

With major improvements in technology, mobile devices and wireless technology have become a part of our lives. As a result of these technological improvements, computing is mobile and ubiquitous these days. All of these technological improvements make it possible to access information of any kind anywhere and at any time as a result of a new research field, ubiquitous computing [1]. For applications running in dynamic environments, context awareness is an important step forward for adapting to dynamic situations [2].

The design and development of an artificial neural network (ANN) for similarity value calculation is proposed in this paper. This neural network is used by the neural agent, which is added to an intelligent, context-aware system (iConAwa), further extending it. iConAwa [3] is an intelligent, multiagent, context-aware system. iConAwa is capable of proactively providing users with context-aware information and services. By utilizing iConAwa, mobile users are able to access to information and services about nearby points of interest (attraction points) (POIs) with respect to their context and by exchanging messages, users can communicate with each other. A POI can be a restaurant, an activity place, movie theater, cultural place, museum, store, etc. The neural agent described in this paper, using an ANN with respect to the user's current context, provides similarity values of attraction points to the context agent.

With the context and the POI ontology, the context and point of interest, respectively, are modeled using Web ontology language (OWL). Ontology refers to the subject of existence. In information science, it is the

*Correspondence: ozgun.yilmaz@ege.edu.tr

shared conceptualization of concepts, properties, and relationships in a specific domain [3,4]. For its advantages, such as sharing information, reusing data, and logical reasoning, an ontology-based context and POI models are used [3,5].

Context-aware applications run in open world and dynamic environments. They require automatic decision making and action taking. Software agent technology is used to cope with these challenges. Software agents are software entities that behave autonomously to reach their goals in an environment [6]. Their running environment is similar to that of context-aware applications. Being autonomous, reactive, and proactive makes software agents suitable for use in context-aware systems, and it makes coping with real-world and dynamic situations easy. Therefore, iConAwa is a multiagent system.

There are three types of users in iConAwa. System administrators are the privileged users who are responsible for the configuration, management, and reliable operation of the system on the server side. Mobile users are the regular users, each having a client agent running in the mobile device. These users are the actual users of the system. They use iConAwa to get context-aware information. The last user type is service provider users. Each of this type of user owns a service agent running in a remote computer and provides services (booking, selling tickets, etc.) to the mobile users.

The significance of iConAwa is that it is a context-aware multiagent system capable of reasoning. With the addition of a neural agent, which uses a neural network, the learning capability is introduced to iConAwa. System administrators, who maintain their own copies of iConAwa, can override and adapt the specification of similarity values according to their own needs. In order to derive high-level contextual information, rule-based context reasoning is performed over the context ontology. iConAwa has expert system properties and learning capabilities, including intelligent agents with proactive and autonomous behavior. As an all-around intelligent system, the iConAwa system described in this paper is unique because of the combination of all the features mentioned earlier.

The contribution of this paper is the use of an ANN to calculate the similarity value for a user's context with regards to a POI. The calculation of similarity values is important because POIs are ranked, sorted, and shown to the user according to similarity values. Previously, similarity value was calculated as equal to the number of keywords (describing the POI) that matched the user's interests, and the calculation of similarity values was hard-coded into the system. Therefore, the similarity metric was fixed. With the introduction of the approach addressed in this paper, system administrators can use their own similarity metric that takes into account their own requirements, further improving the iConAwa system. The system can adapt to needs of different people.

Java Agent DEvelopment Framework was used (http://jade.tilab.com/documentation/tutorials-guides/introduction-to-jade/) to develop iConAwa agents. For using ontologies and inference, the semantic web framework Jena was used (http://jena.apache.org/). Protégé, an ontology editor, was used to develop context and point of interest ontologies [3] (http://protege.stanford.edu).

The subsequent sections of this paper are structured as follows: in Section 2, related work is surveyed and the iConAwa system is evaluated accordingly. In Section 3, an extended iConAwa system is described. In Section 4, the neural network used by the neural agent is proposed. In Section 5, the neural network is evaluated with respect to other possible configurations. In Section 6, the conclusion is discussed.

2. Related work

In this section, previous papers and studies related to this subject are reviewed as follows: CONON [4], Gulliver's Genie [7], COMPASS [8], creation of user-friendly mobile services personalized for tourism (CRUMPET) [9], a context-aware middleware [10], HyCoRE [11], and a context-aware search application [12] are surveyed. Finally, the evaluation and the comparison of iConAwa system and the surveyed works are completed.

2.1. CONON

Obtaining high-level context from low-level contextual information by using reasoning is proposed in CONON [4]. CONON stands for context ontology and is encoded in OWL. Location, user, activity, and computing environment constitute the context. The upper ontology and the domain-specific ontology are combined to obtain the context. The consistency of the contextual information is verified and the high-level implicit context is derived from the low-level implicit context using the logical reasoning over the ontology model [3,4].

In this related work, the conclusions arrived at are as follows: the ontology-based context models necessary for modeling the context and providing reasoning support are possible.

2.2. Gulliver's Genie

Gulliver's Genie is a context-aware mobile tourist system. The mobile user is provided with text, audio, and video content based on the user's location, time, and direction. While showing the user the POI's location, the context model, which is not ontology-based, is also updated simultaneously. The server, using a precaching strategy, allows the user to view the presentations (information) about the POIs on his/her smart phone. While the mobile phone user is driving or walking toward a certain location, the user's destination is predicted and the related data are downloaded to the mobile device before the user's arrival at the destination [3,7,13].

Gulliver's Genie is a multiagent system, and a belief desire intention (BDI) architecture is used where the intelligence of the system comes from reasoning through BDI agents. The presentations are generated considering the user model and other contextual information, as well as the presentation entities being prioritized for downloading. However, this prioritization process is hardcoded into the system and cannot be overridden by the administrator [7].

2.3. COMPASS

By using the WASP platform, supporting context-aware application in web services, and encoding service descriptions in OWL, COMPASS informs the mobile user of available services in the area, such as restaurants and tourist attractions. This is done by communicating with relevant context services. In COMPASS, the derivation of high-level implicit context is done using domain-specific rules. However, it is not clarified if it is rule-based reasoning or not. Ontology models are used to model the context and POIs, and the information presented to the user on the smart phone is simultaneously updated [3,8,13,14].

In COMPASS, the relevance of POIs is predicted by a recommendation service, where similarity values are predicted and are scored by the neural agent. COMPASS uses several prediction methods, such as social filtering [15], case-based reasoning, and category learning [14,16].

2.4. CRUMPET

CRUMPET was developed as a European Union project to help users to locate nearby restaurants, museums, hotels, hospitals, or any POI. As a multiagent system, CRUMPET does not utilize reasoning or the general

context ontology. This similar study also proposes a context-aware support for smart phones, helping them learn users' POIs based on the previous interactions with the user as in CRUMPET [3,9,13,17].

In this related work, an agent-based tourist guide that supports mobile users is proposed. In CRUMPET, an adaptive user model learns user interests from the user's interaction with the system by increasing the interest amount of the object as the user asks for more information about it. Moreover, users' movements are evaluated to infer their interests [3,9,13,17].

2.5. Context-aware middleware

In this related work, a context-aware middleware for providing an automatic home service is proposed. The context-aware middleware consists of an appliance controller that provides communication between devices in the context-aware middleware, a context-aware agent, and a scalable browser [10].

This related work presents a context-aware middleware that uses a neural network to provide home services automatically according to the user's preferences. The middleware is agent-based, but it lacks an ontology-based context model and reasoning.

2.6. HyCoRE

In this related work, the authors deduce that a hybrid artificial intelligence approach is needed in context-aware applications, and they design and develop a foundation for implementing a generalized hierarchical hybrid context reasoning engine for pervasive applications. They argue that this generalized hierarchical hybrid reasoning engine can improve new or existing frameworks lacking sufficient reasoning capabilities [11].

HyCoRE has not been implemented yet. The authors discuss reasoning approaches that can be used for reasoning in different levels and types of context. The ANN approach can be used in applications that involve predicting contexts over numerous possibilities, where there are sufficient time and training data [11].

2.7. A client-server architecture for context-aware search application

In this paper, a client-side context-aware search application is presented. This application is built on a context-aware architecture. The context-aware architecture collects the mobile user's context information, derives the mobile user's current context, distributes the user's context among context-aware applications, and supports context-aware applications. Virtual frame and reference frame concepts are introduced to improve the efficiency on the client side. The virtual frame stores the weights of the context entities that capture the context of the mobile user at specific time frames. The reference frame averages the most-recent N virtual frames, depending on the configuration of individual mobile devices. An ANN is used to calculate the weights for the context entities [12].

2.8. iConAwa

The iConAwa system, with the addition of the neural agent described in this paper, is context-aware and agent-based as well as including context ontology, POI ontology, and rule-based reasoning. Low-level contextual data are used to obtain high-level implicit context by applying rule-based reasoning over context ontology. By doing this, the source code of the system is decoupled from context reasoning, and the similarity values of POIs are calculated using the neural network. Thus, the system has learning capability, and the calculation of similarity values can be overridden by the system administrator. Employing a combination of the features above, the iConAwa system is original since it differs from the previous works.

2.9. Evaluation

The comparable studies are summarized and compared to iConAwa in this section (Table 1). As a context-aware application, Gulliver's Genie [7] is agent-based and uses reasoning, but not an ontology-based context model. COMPASS [8] is a context-aware application, just like the Genie, but it is not agent-based and uses ontology-based context and POI models. It is also not clear whether it uses rule-based reasoning or not. Another context-aware application, CRUMPET [9], is also agent-based, but the user model is ontology-based. Furthermore, CRUMPET does not use context reasoning and does not include an ontology-based context model.

Table 1. Comparison of iConAwa with other related work.

Related work	Agent-based	Context ontology	Reasoning	Learning using ANN
CONON [4]	Not applicable	Yes	DL/Rule-based	Not applicable
Gulliver's Genie [7]	Yes	No	Yes	No
COMPASS [8]	No	Yes	Yes	No
CRUMPET [9]	Yes	No	No	No
Context-aware middleware [10]	Yes	No	No	Yes
HyCoRE [11]	No	Yes	ANN	Yes
Context-aware search application [12]	No	No	ANN	Yes
iConAwa	Yes	Yes	Rule-based	Yes

The related work that has been mentioned up to this point is not capable of machine learning. Reference [10] presents a context-aware middleware that uses a neural network to provide home services automatically according to the user's preferences. This middleware is agent-based, but it lacks an ontology-based context model and reasoning.

Korel and Koo [18] suggest that ANNs can be used for clustering sensor data and context reasoning. HyCoRE [11] and the context-aware search application [12] use ANNs in context reasoning, deriving high-level context from low-level contextual data.

HyCoRE [11] and the context-aware search application [12] are different from iConAwa in the sense that an ANN is used in context reasoning. iConAwa uses rule-based reasoning (inference) to deduce high-level context from low-level contextual data, and an ANN is used for matching user context with POIs. Both of these related works are not agent-based. Only HyCoRE is ontology-based.

The contribution of this work is that, by adding a neural agent that uses an ANN to iConAwa, the system gains the ability to learn, allowing the system owners to adapt the calculation of similarity values of attraction points according to their own needs. The similarity value calculation is not hardcoded into the system, and the administrators can introduce their own similarity metric by providing their own training data.

3. iConAwa

3.1. System architecture

In this section, the extended system architecture of iConAwa [3] is presented. The extended system architecture is shown in Figure 1.

On the server side, a server computer, context agent, neural agent, and directory facilitator (DF) agent are present. Context and POI ontologies are stored in the server.

The client side includes a mobile device equipped with a global positioning system and a client agent

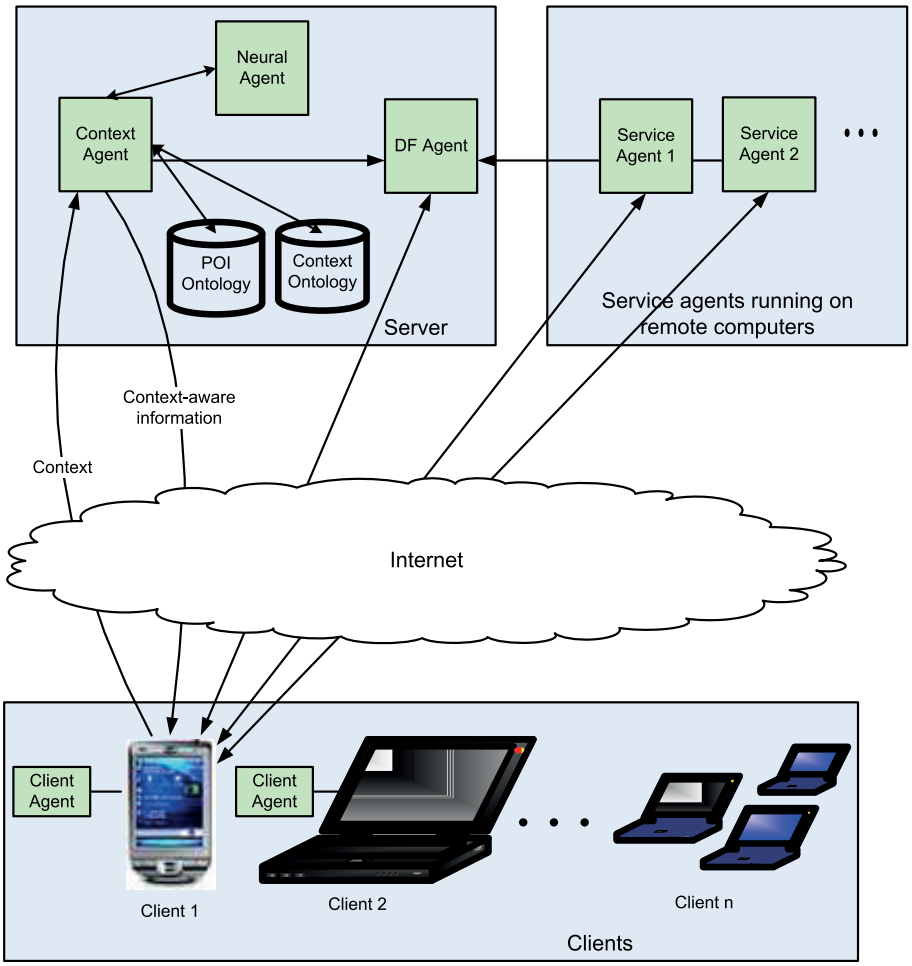

Figure 1. System architecture of the extended iConawa.

running a user interface. In addition, there are service agents running on remote computers. Service agents are specific to a POI and run on remote computers. Users can get services by interacting with the service agents [3].

To use iConAwa, a mobile user starts the application and logs in to the system. Each user is associated with a client agent present in the mobile device. As the user's context changes, the client agent delivers contextual information via messages to the context agent. The context agent forms a user's context by composing raw contextual information. Then the context agent replies by transmitting context-aware information. This reply message is composed of related POIs, their descriptions, and their similarity values, which are obtained from the neural agent. POIs are sorted according to their similarity values. The user can review these points on a map.

In addition, the mobile user's social context is obtained from raw context data by inference. Social context is composed mainly of other nearby users, and nearby users can exchange messages.

The neural agent calculates the similarity value of a POI with respect to the user's current context using an ANN described in Section 4. The neural agent feeds input data acquired from the context agent to the neural network, and then sends the similarity values back to the context agent. More detailed information about iConAwa can be found in reference [3].

3.2. Interactions between the agents

In this section, interactions between the agents are described by a sequence diagram (Figure 2).

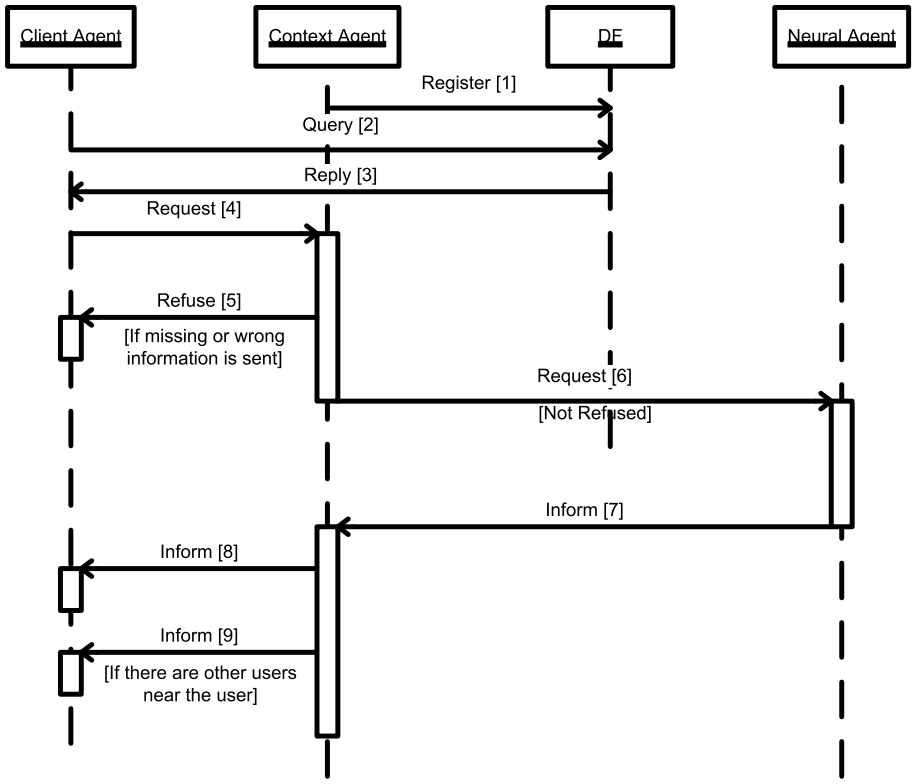

Figure 2. Interactions of the extended iConAwa agents.

Interactions of the agents are listed below:

1. By stating "Contextaware" as the service description, the context agent registers to the DF agent.
2. The client agent queries the DF agent for the "Contextaware" service description.
3. The DF agent replies to the client agent's query. Now the client agent knows the context agent's identifier.
4. A request message is sent to the context agent. This message envelopes login and context information.
5. If the request message contains erroneous information, then a refuse message is forwarded to the client agent that states the error.
6. If the received request is sound, then the context agent sends a request message to the neural agent. The request message contains input values for simulating the neural network.
7. The neural agent sends an inform message to the context agent that contains the similarity value.
8. The context agent informs the client agent by sending context-aware information.
9. If there are any nearby users, then the user agent identifiers of these nearby users are sent to the client agent.

4. The neural network component

Learning improves the success of an intelligent system and strengthens the system's notion of being intelligent. One of the most widely used approaches to learning is ANNs. Basically, an ANN is a primitive model of interconnected nerve cells of the human brain [19].

The neural agent makes use of a neural network to calculate similarity values of attraction points with respect to a user's current context. The neural network is a feed-forward network and supervised learning is used, in which the neural network is trained using a previously created training set.

The calculation of similarity values is important in iConAwa because points of interest are ranked, sorted, and shown to the user according to similarity values. Then the user can get further information and services by clicking a POI. Previously, similarity value was calculated as equal to the number of keywords describing the POI that match the user's interests, and the calculation of similarity values was hard-coded into the system. The contribution of using a neural network for similarity value calculation is that the system can adapt to needs of different people. A system administrator can customize and override the determination of similarity values by providing his/her own training data.

The neural network used by the neural agent was implemented using MATLAB (http://www.mathworks.com/products/matlab/index.html). MATLAB is one of the most used pieces of software in the ANN community. The reasons for choosing MATLAB are as follows: complete software, open-source, powerful and fast processing, easy programming, flexibility, and availability of different neural network models [20]. MATLAB is also more powerful than other ANN frameworks for Java, because of its flexibility and availability of many different neural network models. MATLAB provides MATLAB Builder for Java, which is an extension to the MATLAB compiler, and it can be used to wrap MATLAB functions into one or more Java classes that make up a Java class or package. Each MATLAB function is encapsulated as a method of a Java class and can be called from within a Java application [21].

A wrapper Java class for using the neural network was created using MATLAB Builder for Java. This Java class is used within iConawa system to exploit the neural network. The structure of the neural network is shown in Figure 3. The neural network has 3 layers: the input layer, the hidden layer, and the output layer. In the input and hidden layers, the hyperbolic tangent sigmoid transfer function is used. In the output layer, the linear transfer function is used because the output value is intended to be an unbounded value. The neural network has 9 inputs and 1 output. Inputs are as follows:

- Type of the point of interest (0, 1, 2, 3, . . .)
- Distance (distance in kilometers)
- Keyword describing the POI (0, 1, 2, 3, . . .)
- Does the user's profile include the keyword? (0/1)
- Age (numeric value)
- Gender (0/1)
- Occupation (0, 1, 2, 3, . . .)
- Is the POI open at the time? (0/1)
- Education (0, 1, 2, 3, . . .)

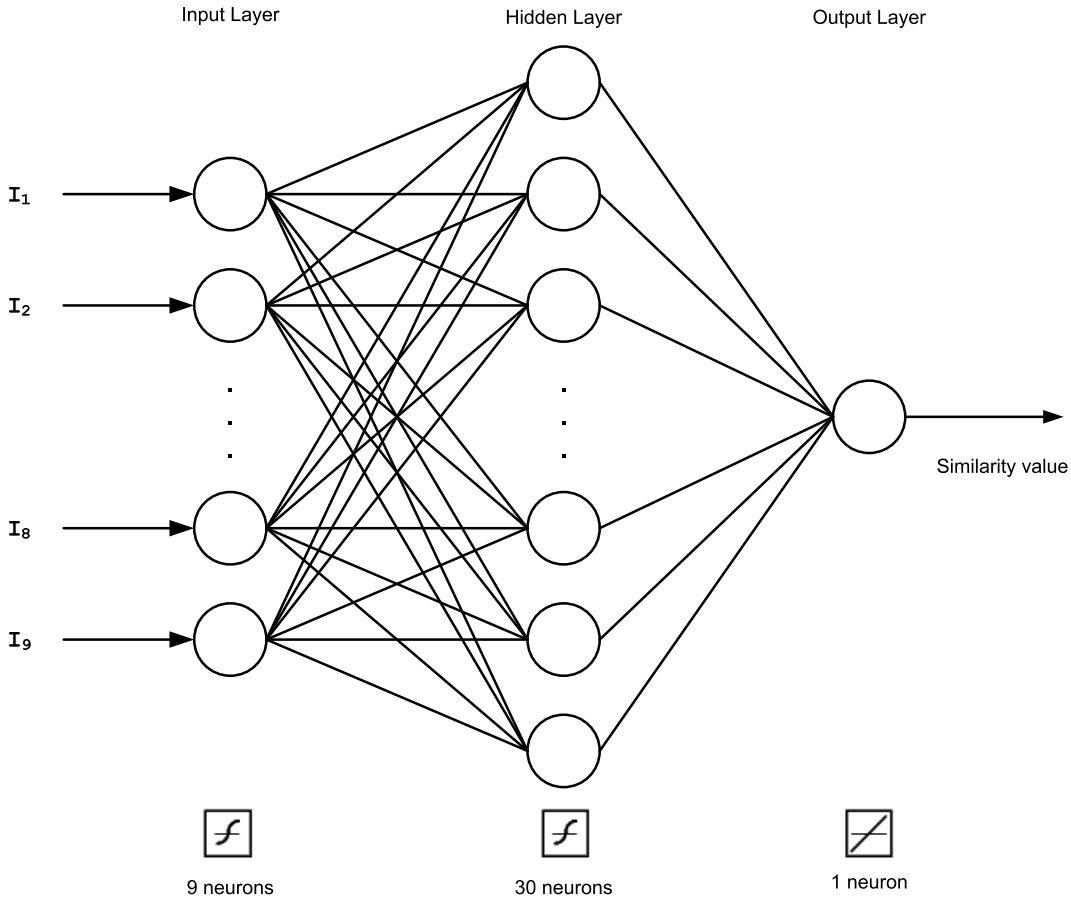

Figure 3. The structure of the neural network.

The output is the similarity value, which is an unbounded numeric value. If the similarity value of a certain POI type is not related to one or more of the inputs, then a constant value for these inputs can be used.

As a POI can have more than one keyword, the neural network is simulated for every keyword separately. Then the mean value of the outputs obtained for each keyword is calculated and the mean value is the similarity value.

The hidden layer has 30 neurons. The use of the hidden layer in the network decreases the time to train the network and the mean square error (MSE) of the network, which is further discussed in the next section.

5. Results and discussion

There is no easy way to determine the optimum number of neurons for the hidden layer. In this study, the optimum number of neurons in the hidden layer was determined by trial and error. A benchmark was conducted using different layer sizes on a Pentium IV 3.2 GHz computer with 1 GB of RAM running Windows XP Professional.

First, each neural network was trained for 200 trials using a training set consisting of 128 inputs. Then the neural networks were evaluated according to their training time and mean square errors. The maximum epoch number for training was 10,000. The Levenberg–Marquardt backpropagation training method was used to train the network, and the training was completed when the minimum gradient value or the maximum epoch count was reached. The results are shown in Table 2.

Table 2. Training performances of different neural network configurations.

No. of neurons in the hidden layer	Total epoch count	Mean epoch count	Total time (s)	Mean time (s)	Total MSE	Mean MSE
No hidden layer	1,686,588	8432.94	20,054.52	100.2723	2.16E+02	1.08E+00
10	575,115	2875.575	14,963.8125	74.819	9.01E+01	4.50E-01
15	57,641	288.205	2237.796875	11.189	6.01E-13	3.01E-15
20	9086	45.43	490.515625	2.452	5.57E-21	2.79E-23
25	6621	33.105	467.4375	2.337	3.66E-21	1.83E-23
30	4721	23.605	427.4375	2.137	4.12E-21	2.06E-23
35	4172	20.86	462.5	2.312	2.34E-21	1.17E-23
40	4034	20.17	573.09375	2.865	3.66E-21	1.83E-23
45	3645	18.225	617.828125	3.089	5.39E-21	2.69E-23
50	3254	16.27	648.140625	3.24	2.22E-21	1.11E-23

All of the networks listed in Table 2 are acceptable when they are evaluated according to their mean square errors, except the network with no hidden layer and the network with 10 neurons in the hidden layer, which had high maximum mean square error values.

Table 2 shows that, as the number of neurons in the hidden layer increases, the total and the mean epoch count to train the networks decrease. In the case of training time, it is a little different. Although the epoch count decreases continuously, the total and mean time values decrease up until 30 neurons. After 30, the total and mean time to train the networks begin to increase, because as the number of neurons in the hidden layer increases, the complexity of the network increases and, as a result, it takes more time to train the network.

In this study, the neural network with a hidden layer of 30 neurons was chosen because it had the lowest total and mean training times. Figures 4 and 5 show the epoch count and training time values, respectively, across 200 trials for the chosen network.

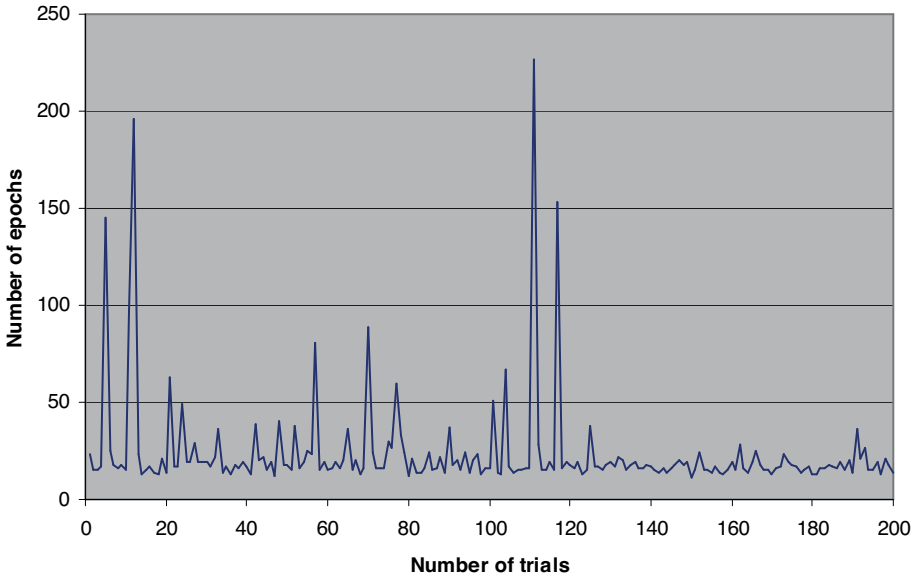

Figure 4. Number of epochs to train the neural network across 200 trials.

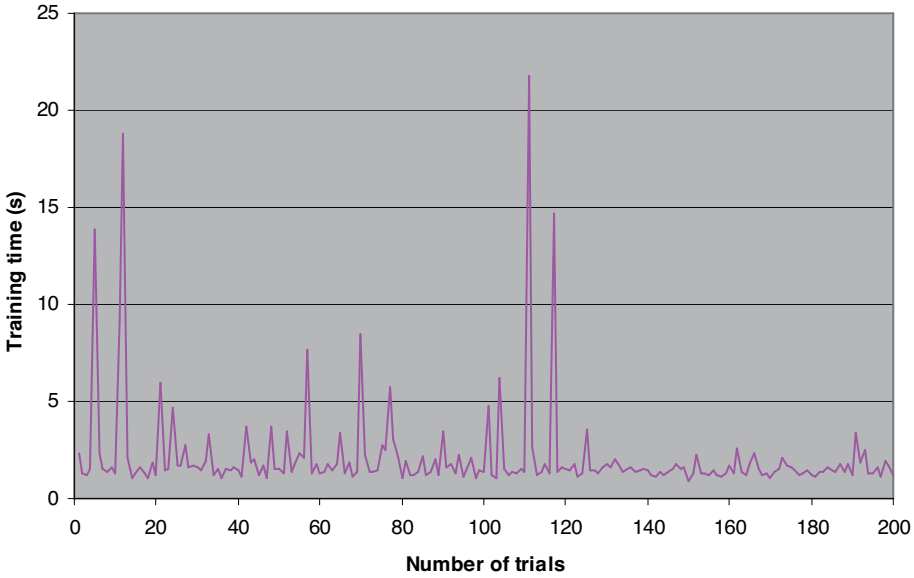

Figure 5. Amount of time to train the neural network across 200 trials.

6. Conclusion

In this paper, the development of an ANN for calculating similarity values in a context-aware system is described. The neural network is used by the neural agent of iConAwa to provide the context agent with the similarity values of POIs with respect to the user's current context. This is important because POIs are ranked, sorted, and shown to the user according to similarity values.

The contribution of this paper is the use of an ANN to calculate the similarity value for the user's context with regards to a POI. With the introduction of the approach addressed in this paper, system owners can use their own similarity metric considering their own requirements, which further improves the iConAwa system. The system can adapt to needs of different people.

Alternative neural network structures were evaluated and the network with superior training performance was chosen. With the addition of the neural network, iConAwa gains the ability to learn, and the system combines expert system characteristics with the ability to learn. Learning strengthens the notion of being an intelligent system, and the calculation of similarity values can be adapted to different system owner's personal preferences.

In development and evaluation of the neural network, the MATLAB environment was used. In addition, MATLAB's Builder for Java was used to create a wrapper java class to use the trained network from the Java code.

References

[1] Chen G, Kotz D. A survey of context-aware mobile computing research. Tech Rep 2000: TR2000-381. Hanover, NH, USA: Dartmouth College.

[2] Belotti R, Decurtins C, Grossniklaus M, Norrie MC, Palinginis A. Modelling context for information environments. In: Baresi L, Dustdar S, Gall HC, Matera M, editors. Ubiquitous Mobile Information and Collaboration Systems. Berlin, Germany: Springer, 2005, pp. 43-56.

[3] Yılmaz Ö, Erdur RC. iConAwa - an intelligent context-aware system. Expert Syst Appl 2012; 39: 2907-2918.

[4] Wang XH, Zhang DQ, Gu T, Pung HK. Ontology based context modeling and reasoning using OWL. In: Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops; 14–17 March 2004; Washington, DC, USA: IEEE. pp. 18-22.

[5] Gu T, Wang XH, Pung HK, Zhang DQ. An ontology-based context model in intelligent environments. In: Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference; 2004; San Diego, CA, USA. pp. 270-275.

[6] Wooldridge M. An Introduction to Multiagent Systems. Hoboken, NJ, USA: John Wiley & Sons Ltd, 2002.

[7] O'Grady MJ, O'Hare GMP. Gulliver's Genie: agency, mobility & adaptivity. Comput Graph 2004; 28: 677-689.

[8] Van Setten M, Pokraev S, Koolwaaij J. Context-aware recommendations in the mobile tourist application COMPASS. In: Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops; 2004; IEEE Computer Society. pp. 235-244.

[9] Schmidt-Belz B, Poslad S, Nick A, Zipf A. Personalized and location-based mobile tourism services. In: Proceedings of the Mobile HCI '02 with the Workshop on Mobile Tourism Support Systems; 2002.

[10] Choi J, Shin D, Shin D. Research and implementation of the context-aware middleware for controlling home appliances. IEEE T Consum Electr 2005; 51: 301-306.

[11] Beamon B, Kumar M. HyCoRE: Towards a generalized hierarchical hybrid context reasoning engine. In: Proceedings of the 2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops; 29 March–2 April 2010; IEEE. pp. 30-36.

[12] Gui F, Guillen M, Rishe N, Barreto A, Andrian J, Adjouadi M. A client-server architecture for context-aware search application. In: Proceedings of the 2009 International Conference on Network-Based Information Systems; 19–21 August 2009; IEEE. pp. 539-546.

[13] Schwinger W, Grün C, Pröll B, Retschitzegger W, Schauerhuber A. Context-awareness in mobile tourism guides - a comprehensive survey. Tech Rep 2005. Linz, Austria: Johannes Kepler University.

[14] Pokraev S, Koolwaaij J, Van Setten M, Broens T, Costa PD, Wibbels M, Ebben P, Strating P. Service platform for rapid development and deployment of context-aware, mobile applications. In: Proceedings of the 2005 IEEE International Conference on Web Services; 11–15 July 2005; Washington, DC, USA: IEEE. pp. 639-646.

[15] Shardanand U, Maes P, Social information filtering: Algorithms for automating "word of mouth." In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '95; 1995. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co. pp. 210-217.

[16] Van Setten M. Experiments with a recommendation technique that learns category interests. In: Proceedings of the 2002 IADIS International Conference WWW/Internet; 13–15 November 2002; Lisbon, Portugal. pp. 722-725.

[17] Poslad S, Laamanen H, Malaka R, Nick A, Buckle P, Zipl A. Crumpet: creation of user-friendly mobile services personalised for tourism. In: Proceedings of the 2nd International Conference on 3G Mobile Communication Technologies; 26–28 March 2001; London, UK. pp. 28-32.

[18] Korel BT, Koo SGM. A survey on context-aware sensing for body sensor networks. Wirel Sens Netw 2010; 2: 571-583.

[19] Negnevitsky M. Artificial Intelligence: A Guide to Intelligent Systems. 2nd ed. Boston, MA, USA: Addison Wesley, 2005.

[20] Baptista D, Abreu S, Freitas F, Vasconcelos R, Morgado-Dias F. A survey of software and hardware use in artificial neural networks. Neural Comput Appl 2013; 23: 591-599.

[21] MathWorks, Inc. MATLAB Builder for Java - The Language of Technical Computing - User's Guide Version 1, 2006. Natick, MA, USA: MathWorks.