Hindawi
Mobile Information Systems
Volume 2018, Article ID 6920679, 9 pages
https://doi.org/10.1155/2018/6920679

Research Article

Automatic Task Classification via Support Vector Machine and Crowdsourcing

Hyungsik Shin [1] **and Jeongyeup Paek** [2]

[1]*School of Electronic and Electrical Engineering, Hongik University, Seoul, Republic of Korea*
[2]*School of Computer Science and Engineering, Chung-Ang University, Seoul, Republic of Korea*

Correspondence should be addressed to Jeongyeup Paek; jpaek@cau.ac.kr

Received 5 December 2017; Revised 14 March 2018; Accepted 4 April 2018; Published 2 May 2018

Academic Editor: Dingqi Yang

Automatic task classification is a core part of personal assistant systems that are widely used in mobile devices such as smartphones and tablets. Even though many industry leaders are providing their own personal assistant services, their proprietary internals and implementations are not well known to the public. In this work, we show through real implementation and evaluation that automatic task classification can be implemented for mobile devices by using the support vector machine algorithm and crowdsourcing. To train our task classifier, we collected our training data set via crowdsourcing using the Amazon Mechanical Turk platform. Our classifier can classify a short English sentence into one of the thirty-two predefined tasks that are frequently requested while using personal mobile devices. Evaluation results show high prediction accuracy of our classifier ranging from 82% to 99%. By using large amount of crowdsourced data, we also illustrate the relationship between training data size and the prediction accuracy of our task classifier.

1. Introduction

Artificial intelligence and machine learning has received much attention in our information technology era, and we are observing more and more applications in our daily lives than before. In particular, many industry leaders have developed and introduced top-notch applications based on artificial intelligence [1–5]. These applications include personalized content recommendations and personal assistant services [3–7].

Many advanced personal assistant services heavily depend on natural language understanding (NLU) for human-computer interactions [8–10]. There are also many systems that are based on touch-driven interactions [11, 12]. Nowadays, many machines can interact with humans with a certain level of intelligence, and at the core of them are artificial intelligence algorithms and natural language processing.

There are many unsolved problems of natural language understanding, and the problem of automatically classifying a given natural language input into a suitable task or category is one of them. Many researchers and industry leaders have suggested various algorithms and approaches to tackle the problem [8, 9, 13–20]. These research and development activities later resulted in various personal assistant services such as Apple's Siri, Google's Google Now, and Amazon's Alexa.

However, the personal assistant services that are provided by the industry leaders are proprietary, and their internals and implementations are not well known to the public. As they have been continuously updated and improved over the past several years, we believe that their implementations are highly sophisticated and complicated combinations of many different algorithms and the state-of-the-art technologies. Therefore, we asked ourselves the following question: "Is it possible to implement a personal assistant system that is simple enough to be built by applying a well-known machine learning algorithm and personally crowdsourced data?" By answering this question, we hope that our work motivates many researchers and small

industries to build their own intelligent systems in their particular domains.

With the motivation in mind, in this paper, we introduce our own implementation of an automatic task classification system, which is based on a classical machine learning algorithm and crowdsourcing. Many different classification algorithms have been proposed and introduced to the artificial intelligence and machine learning community. To implement our task classification module, we used the support vector machine (SVM), a popular classification algorithm. In particular, we used the LibShortText library [21], which is an extension of Liblinear [22], a library implementing linear support vector machine algorithms.

Using our implementation, we show that the support vector machine algorithm can be successfully used for building personal assistant services, in particular, task classifiers for mobile devices. This task classifier can take a natural language text input and classify the input text into an implied task category among many predefined tasks. Therefore, it can understand humans' natural language command and execute the intended task accordingly on behalf of the user.

Even though Apple, Google, and Amazon are not disclosing the internal architecture or algorithms that were used to implement their own personal assistant services [3], it is believed that they are making use of a large amount of data that they have collected from various sources to implement their systems. In order to train our classification module, we also collected our own training data, and we describe how we collected our data via crowdsourcing.

By using a large amount of collected training data, we investigate and present a relationship between task classification accuracy of our classifier and training data size. We verify that the more training data we use, the better prediction accuracy we can get, but the performance increase rate drops.

This paper is organized as follows. Section 2 introduces a couple of commercial personal assistant systems and the support vector machine algorithm. Then, an open-source library implementing the support vector machine is briefly introduced. Our classifier uses the library to build a task classifier model. In Section 3, we describe our classifier and the library on which the classifier is built. Section 4 describes our crowdsourcing procedure for collecting training data that are used to train our classifier model. Section 5 shows that our classifier can classify short English texts into implied task categories. In particular, precision and recall values are presented for each task. The relationship between prediction accuracy and training data size is also investigated. In Section 6, we propose an overall architecture of a possible implementation of a personal assistant system, which is based on our task classifier. Finally, Section 7 concludes the paper.

2. Background and Prior Work

Before we propose our task classifier, we introduce some prior work on natural language processing and a couple of personal assistant systems. Then, a brief background on the support vector machine algorithm is introduced.

2.1. Natural Language Processing. Natural language processing is a fairly large research area and has a long history in the computer science community. The following are a few prior work in the field.

In 1972, Winograd tried to implement a computer system that can interact with human beings in English [14]. Kuhn and De Mori tried a new data structure, semantic classification tree, to implement a building block for robust matchers for NLU tasks [9]. Manning and Schütze describe statistical natural language processing in their book [8]. Yi et al. presented a sentiment analyzer that extracts sentiment about a subject using natural language processing techniques [15]. Collobert et al. proposed a unified neural network architecture and learning algorithm that can be applied to various natural language processing tasks including part-of-speech tagging, chunking, named entity recognition, and semantic role labeling [16].

Task or category classification has much prior work, too. In 1994, Cavnar and Trenkle proposed a text classification approach based on N-gram [17]. Yang and Pedersen performed a comparative study on feature selection methods in text categorization [18]. Text categorization via SVM was studied by Joachims in 1998 [19]. Yang and Liu performed a study on five different methods for text classification [23]. Sebastiani summarized many different approaches for text classification based on machine learning [24]. Pang et al. performed a study on sentiment analysis (positive or negative) by employing three different machine learning algorithms: naive Bayes, maximum entropy classification, and support vector machine [25]. Tong and Koller proposed support vector machine active learning for text classification applications [20]. Leopold and Kindermann showed that term-frequency transformations have a larger impact on the performance of SVM for text classifications than the kernel functions [26]. Genkin et al. proposed a new approach based on logistic regression that can handle high-dimensional data such as natural language text [27]. Lan et al. proposed a new term weighting method for text classification [28].

2.2. Personal Assistant Systems. One of the innovative and well-known automatic task classification systems for a natural language input sentence would be Siri, a personal assistant system developed by Apple, Inc. [29]. Soon after Siri was introduced, Google also started providing its own similar service, which is called as Google Now [30, 31].

These two systems can understand humans' natural language command, which means that they can classify a given natural language input to a command implied by the input text and perform the predicted task accordingly. For example, these systems understand a voice input command such as "Call John," and on behalf of the user, they perform automatically the user's intended task, which is a "Call" task.

However, Apple and Google have not disclosed how these systems are implemented. Therefore, we designed and implemented our own automatic task classifier based on a widely used classification algorithm, the support vector machine.

2.3. The Support Vector Machine. There are many classification algorithms that are well known to the machine learning community. In particular, Deep Learning [32–34] is a very hot topic these days. If designed and trained carefully, deep learning algorithms usually outperform (in terms of classification accuracy) most of the previously known classification algorithms such as the support vector machine, random forests, and naive Bayes in many domains. However, in order to train a competitive deep learning network, a very large amount of training data is usually required. Furthermore, deep neural networks are often regarded as black boxes because it is hard to understand how the networks classify test instances into correct categories through the deep network layers.

On the contrary, the support vector machine is seen to be more interpretable than deep neural networks, and it had been mostly used in many classification problems. Even though there is no one universal algorithm that outperforms every other classification algorithms in various domains, the support vector machine was widely used due to its relatively powerful performance over many different areas [35]. Considering our goal of this work, we decided to use the support vector machine algorithm rather than using more contemporary but complicated deep learning algorithms.

Support vector machine algorithm was conceptually invented in 1963 by Vapnik and Chervonenkis [36]. In 1992, Boser et al. proposed a way to create nonlinear classifiers via the kernel trick [37]. A couple years later, Cortes and Vapnik introduced the concept of the soft margin [38]. Since its introduction, SVM has seen many applications such as hand-written character recognition.

There are many implementations of SVM with different optimization algorithms. Fan et al. implemented an open-source library for large-scale linear classification, which is named as Liblinear [22]. Liblinear supports logistic regression and linear support vector machines. Another open-source library for short text classification and analysis, called LibShortText, was implemented [21]. LibShortText is an extension of Liblinear, and it can train a classification model with a given training data set consisting of short natural language texts with labels.

3. Automatic Task Classifier

Our main idea is that a practical task classifier, a core part of personal assistant systems, can be implemented to reach a sufficient accuracy by using a classical classification algorithm and basic natural language processing techniques. As we have briefly introduced in Section 2, there exists an open-source library for text classification based on the support vector machine. Therefore, we adopted this library to design our task classifier instead of reinventing the wheels.

3.1. Predefined Tasks. We implemented our own automatic task classifier that can classify a given natural language input text to the most appropriate task among the thirty-two predefined tasks. The thirty-two distinct tasks that we have used are shown in Table 1. We picked these tasks based on our observation that they would span most frequently used tasks that a user can command their mobile devices such as smartphones or tablets. However, these thirty-two tasks are not meant to be hardcoded; users can define any task list of their own interest.

Table 1: The predefined thirty-two tasks for mobile devices.

Search		App	Chatting
Transportation	Map	Call	Greeting
Travel	Music	E-mail	Praise
Movie	Photo	Camera	Dispraise
Book	News	Check schedule	Boredom
Game	Apps	Schedule	Love
Restaurant	Recipe	Memo	—
Shopping	Weather	Timer	—
Hospital	—	Alarm	—
Wikipedia	—	Music player	—
Information	—	SNS	—

For example, the thirty-two tasks contain the "Call" task, and our task classifier can classify an input text "Call John." In other words, our classifier will automatically recognize the user's intended task, which is the "Call" task, and the personal assistant system will command the mobile device to search its contacts list to find "John" and finally place a call to him. Even though we defined our own thirty-two tasks targeting mobile devices, different use cases can define their own task lists. For an instance, navigation systems may have totally different tasks such as "Search Location" or "Cancel Navigation."

3.2. Training Task Classifier. In order to implement our automatic task classifier, we exploit LibShortText [21], an open-source library implementing a short-text classifier. This library is very well implemented and provides a capability to change the parameters of the support vector machine algorithm or natural language processing. As the authors of LibShortText claimed, our preliminary experimentation has shown that the default parameters of the library result in very good classification accuracies for our purpose, if not the best, so we mostly followed their recommendations as is.

In addition, in an attempt to enhance the accuracy further, we designed a preprocessing step, which is to replace some words into more general categories. For example, if the given sentence is "I want to have a sushi," then the word "sushi" is replaced with "categoryFood." Therefore, the final sentence in this case becomes "I want to have a categoryFood."

We experimented with the idea of word replacement because replacing more specific words with more general terms may increase classification accuracies by decreasing the dimension of the input feature space (the space of N-grams). In order to implement the preprocessing of word replacement, we first created a dictionary, which maps some words to more general categories. For example, "sushi" and "pizza" are mapped to the "categoryFood" category. After we generated the dictionary, we applied the preprocessing step to the training data set. In other words, all the training data

sentences were transformed to sentences where specific words are replaced with corresponding category titles. Of course, when the task classifier classifies a test input sentence, the same preprocessing step should be performed on the test sentence before the classification process kicks in.

To our disappointment, however, the experiment results were not promising; the classification accuracy with the preprocessing was not higher than the case without that step. We believe there may be many reasons for this. First of all, the classification accuracy of the support vector machine is already very high without the preprocessing, which makes it very hard to increase the accuracy further. Second, the feature space dimension is not reduced enough to affect the classification accuracy.

After we confirmed the experiment results, we chose to stay with the recommended setting of LibShortText without the preprocessing. We also want to mention that the preprocessing takes computation time, which is another reason why we chose not to apply our tested preprocessing.

The LibShortText library requires a training data set to train a support vector machine, so we used a popular crowdsourcing platform to collect our training data set. In order to achieve high classification accuracy for general natural language text inputs, we need a large amount of data. The data collection process is described in the next section.

4. Data Collection via Crowdsourcing

This section describes how we collected our own training data set for our task classifier training.

4.1. Amazon Mechanical Turk. Crowdsourcing has been a powerful way to obtain human intelligent services, ideas, or content by soliciting contributions from a large group of people and especially from online communities [39]. A well-known online survey platform, SurveyMonkey, is a good example of many services that can be used for collecting data via crowdsourcing. Many people are now using SurveyMonkey in small scales for personal, academic, or industrial purpose.

Amazon.com, Inc. is also providing a popular and commercial crowdsourcing platform called Amazon Mechanical Turk (MTurk). MTurk provides an easy-to-use system for collecting a large amount of data sets via crowdsourcing. There have been many research results about the data quality collected by MTurk. Buhrmester et al. described and evaluated the potential contributions of MTurk to psychology and other social sciences [40]. Paolacci et al. addressed potential concerns about the quality of collected data through MTurk by presenting new demographic data about the Mechanical Turk subject population, reviewing the strengths of MTurk relative to other online and offline methods of recruiting subjects and comparing the magnitude of effects obtained using Mechanical Turk and traditional subject pools [41]. Kittur et al. also performed a study on validity of MTurk platform [42]. Many of them indicate that MTurk is a good crowdsourcing platform to collect high-quality data with inexpensive monetary costs.

We collected our own training data set using MTurk to train our task classification engine. MTurk enables crowdsourcing requesters to upload their questionnaires. Once uploaded, MTurk publishes the questionnaires to the MTurk open marketplace so that many MTurk workers can answer the uploaded questionnaires and get paid by the requester.

4.2. Our Data Collection Process. In order to collect English sentences for commanding any of the predefined thirty-two tasks, we created a questionnaire that requests a worker to fill out his/her own example sentence for each different task. For example, for the task of "Restaurant Search", one person can provide an example sentence such as "Find me a good Italian restaurant nearby," and another person can provide a different sentence such as "best Korean food in San Francisco." Once a worker finished filling out an example sentence for each of all the thirty-two tasks, then we compensated the worker for their contribution.

Through the aforementioned MTurk crowdsourcing process, we were able to collect 65,890 sentences for the thirty-two tasks. Each task has at least 2,000 sentences. All these sentences are human generated, so the data set has high quality and variety. For example, a worker provided an example sentence, "I am hungry" for the task of "Restaurant Search," whereas many other workers just provided names of various cuisines such as "Pizza" or "Sushi."

5. Prediction Accuracy

In order to train our classifier and evaluate the classification accuracy, we applied the tenfold cross-validation approach using the 65,890 sentences for the thirty-two tasks collected via crowdsourcing. While applying tenfold cross validation, we measured the precision and recall values for each task.

5.1. Precision and Recall. While performing the tenfold cross validation, the precision and recall values for each task can be computed as follows for each fold.

Suppose that we randomly partition all the collected data set T into ten equal folds. We partitioned T so that each fold has roughly equal number of data points for each task. To compute precision and recall values corresponding to a partition P_i, $i = 1, 2, \ldots, 10$, we form a training data set consisting of all the data except those belonging to P_i. In other words, if we call the training data set for the fold P_i as T_i, we have

$$T_i = \{d \in T \mid d \notin P_i\}. \quad (1)$$

By using T_i as a training data set, we train a task classifier model f_i. We then measure the prediction accuracy of the model f_i using the fold P_i as a validation set.

For each pair of task t_j, $j = 1, 2, \ldots, 32$, and fold P_i, $i = 1, 2, \ldots, 10$, the precision and recall values are computed by the following equations:

Table 2: The precision and recall values of each task as well as the top 3 misclassifications.

Task name	Precision (%)	Recall (%)	Top 3 misclassifications		
Transportation	91.48	94.33	Travel 1.61%	Map 1.46%	Restaurant 0.76%
Map	83.87	82.76	Restaurant 5.53%	Shopping 4.49%	Transportation 3.45%
Travel	93.64	94.44	Transportation 0.85%	Wikipedia 0.81%	Restaurant 0.62%
Music	88.98	86.25	Music player 6.33%	Wikipedia 3.07%	Book 1.56%
Movie	89.74	93.10	Book 1.28%	Information 1.18%	Music 1.09%
Photo	95.86	95.13	Information 0.99%	Camera 0.90%	Wikipedia 0.61%
Book	91.21	91.51	Music 1.52%	Wikipedia 1.47%	Shopping 1.09%
News	93.08	94.14	Wikipedia 1.37%	Information 0.85%	Book 0.61%
Game	88.23	89.07	Apps 6.48%	Music player 0.71%	Movie 0.47%
Apps	90.02	88.92	Game 6.92%	Map 0.47%	Movie 0.43%
Restaurant	86.54	88.35	Map 4.31%	Shopping 2.84%	Transportation 1.14%
Recipe	97.14	98.15	Restaurant 0.43%	Information 0.24%	Map 0.14%
Shopping	86.25	88.75	Map 3.99%	Restaurant 2.71%	Hospital 1.04%
Weather	97.90	99.20	News 0.14%	Information 0.14%	Check schedule 0.14%
Hospital	95.55	94.92	Shopping 1.28%	Map 1.05%	Restaurant 0.81%
Wikipedia	83.05	78.63	Information 7.98%	News 3.32%	Music 1.85%
Information	82.51	83.88	Wikipedia 5.61%	Music 1.24%	Book 1.14%
Call	98.57	98.34	Greeting 0.48%	Love 0.24%	Movie 0.14%
E-mail	99.38	99.19	Call 0.14%	Dispraise 0.14%	Memo 0.10%
Camera	98.16	98.72	Photo 0.24%	Apps 0.19%	Movie 0.14%
Check schedule	90.60	92.97	Schedule 4.66%	Boredom 0.48%	Memo 0.38%
Schedule	93.43	91.35	Check schedule 6.51%	Memo 0.90%	Timer 0.14%
Memo	97.57	97.29	Schedule 0.62%	E-mail 0.29%	Timer 0.24%
Timer	97.59	98.24	Alarm 0.76%	Memo 0.24%	Information 0.14%
Alarm	98.38	98.38	Timer 0.90%	Greeting 0.24%	Movie 0.14%
Music player	91.23	96.28	Music 2.62%	Timer 0.14%	Love 0.14%
SNS	98.69	96.36	News 1.09%	Movie 0.57%	Wikipedia 0.43%
Greeting	93.87	92.14	Dispraise 1.51%	Check schedule 1.11%	Praise 1.00%
Praise	89.12	89.52	Dispraise 2.79%	Love 1.95%	Restaurant 1.06%
Dispraise	84.00	83.48	Boredom 3.29%	Praise 3.07%	Shopping 1.79%

Table 2: Continued.

Task name	Precision (%)	Recall (%)	Top 3 misclassifications		
Boredom	89.76	84.14	Dispraise 4.80%	Game 2.79%	Greeting 1.45%
Love	92.40	88.27	Praise 4.41%	Dispraise 2.12%	Boredom 1.68%

Accuracy of our classifier (in terms of precision and recall) is promisingly high ranging from 82% up to 99%.

$$\text{Precision}(t_j, P_i) = \frac{\left|\left\{d \in P_i \mid f_i(d) = t_j, t(d) = t_j\right\}\right|}{\left|\left\{d \in P_i \mid f_i(d) = t_j\right\}\right|},$$
$$\text{Recall}(t_j, P_i) = \frac{\left|\left\{d \in P_i \mid f_i(d) = t_j, t(d) = t_j\right\}\right|}{\left|\left\{d \in P_i \mid t(d) = t_j\right\}\right|}, \quad (2)$$

where $t(d)$ and $f_i(d)$ are the original and the predicted task labels of the data sentence d, respectively.

5.2. Measured Prediction Accuracy. The tenfold cross-validation results are shown in Table 2. As shown in Table 2, the precision and recall values are very high ranging from 82% up to 99%. Even though there are some misclassifications between similar tasks, the overall prediction accuracy is promising. Therefore, our task classifier can be employed to build practical personal assistant services.

We can also observe that only a few groups of similar tasks have relatively high misclassification ratios. For example, the three tasks of "Map," "Restaurant," and "Shopping" search were mostly confused with each other. The "Music" search and "Music player" app launch tasks were also mostly confused with each other. The "Apps" search and "Game" app launch tasks were mostly confused with each other, and the "Wikipedia" and "Information" search tasks were also confused with each other. Finally, the five chatting tasks were confused among themselves.

In order to increase the accuracy of our task classifier further, we may build a second layer of classification, which is applied to and classifies each group of similar tasks to a more accurate task inside the group. The second layer need not use the support vector machine as the first layer; it may use rule-based classifiers or any other algorithm.

5.3. Accuracy and Training Data Size. We performed an experiment to investigate the relation between training data size and classification accuracy. It is expected that the more the data we use for training, the more the accuracy we can get. We wanted to confirm this expectation and to get the precise relation between the classifier performance and the training data size. For this experiment, we randomly chose 20% of all the collected data points as the test set. Then, we used the remaining 80% of the data set as a training data pool, so that we can sample a certain amount of data points randomly from the pool.

More specifically, suppose that S is the randomly sampled test set whose size is 20% of the collected data set T. Then, the remaining 80% of the collected data is used as the training data pool P, from which we sample different sizes of training data. Therefore, we have

$$P \cup S = T,$$
$$P \cap S = \varnothing. \quad (3)$$

In particular, we tested with ten different sizes of training data: 10%, 20%, 30%, ..., 100% of the training data pool P. For each different training data size of $(i/10)|P|$ for $i = 1, 2, 3, \ldots, 10$, we repeated random sampling from the pool P ten times to train ten different classifier models with the same training data size. In other words, for each i, we train ten different task classifiers f_{ij} for $j = 1, 2, 3, \ldots, 10$ by sampling training data of the same size from the pool P ten times. Then, using each classifier f_{ij}, we measured the classification accuracy acc_{ij} as

$$\text{acc}_{ij} = \frac{\left|\left\{d \in S \mid f_{ij}(d) = t(d)\right\}\right|}{|S|}, \quad (4)$$

where $f_{ij}(d)$ and $t(d)$ are the predicted and original task of the input text d, respectively. Therefore, we have ten different accuracy values for each training data size.

The results of the experiment are shown in Figure 1, where a boxplot is generated with ten different accuracy values for each different training data size. The figure shows that the classifier performance increases as the training data size increases, but the performance increase rate drops as the training data size increases. This result is reasonable and verifies our original hypothesis.

6. Personal Assistant Systems

So far, we have described our proposed task classifier that is based on the support vector machine algorithm. The proposed task classifier may be used to build a personal assistant system. Figure 2 shows the overall architecture of a possible implementation of a personal assistant system. A user's voice command is first transmitted to a server, which is running the speech-to-text converter and the task classifier.

For the speech-to-text converter, any suitable converter may be used; for example, we may use Sphinx speech recognition system [43–45].

The received speech at the server is converted to text, and the converted text is given to the task classifier, a core part of the overall system. The task classifier classifies the given text into the most probable task, and the predicted task is transmitted back to the user's mobile device. Then, the task launcher of the personal assistant system performs the predicted task accordingly on behalf of the user.

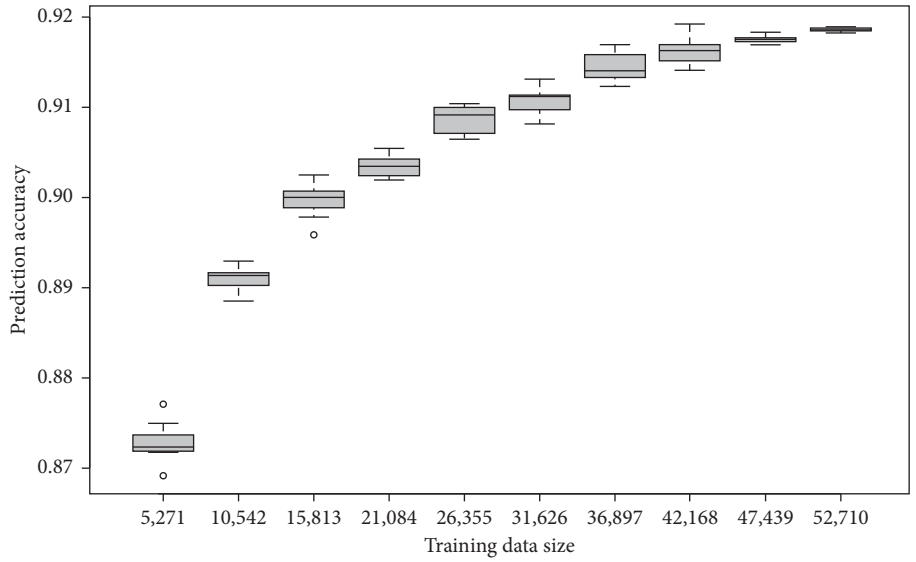

Figure 1: Prediction accuracy increases as we use more training data, but the increase rate drops.

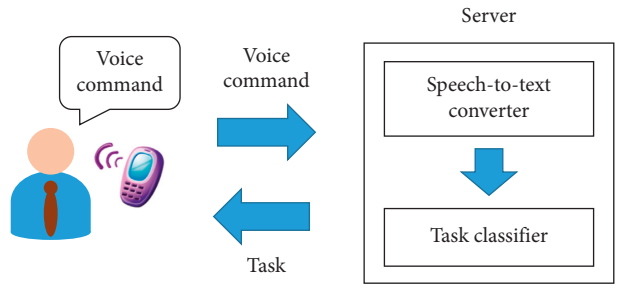

Figure 2: A proposed architecture of a personal assistant system.

Since the proposed task classifier is built upon the linear support vector machine, the computation time typically ranged from a few milliseconds to a few tens of milliseconds. Of course, as we use more powerful servers, we can reduce the computation time more.

7. Conclusion

We presented a method to implement personal assistant services that can understand human's natural language commands. Even though there already exist such services including Siri, Google Now, and Alexa, the internal technologies have not been disclosed and are not well known to the public. Therefore, we investigated whether it is possible to build a task classifier, a core part of personal assistant services, using a well-known machine learning algorithm. Our implementation is based on the support vector machine, a widely used classification algorithm in many domains.

To train our support vector machine with sufficient data, we collected our own training data set by using a popular crowdsourcing platform, the Amazon Mechanical Turk. We predefined thirty-two tasks that are frequently commanded while using mobile devices and collected natural language sentences for each task by using the crowdsourcing platform. Through this process, we were able to collect 65,890 natural language sentences in total.

We tested our task classifier performance with the tenfold cross-validation approach. The evaluation results show that the precision and recall values of our classifier are very high. This result indicates that our simple approach can be employed to implement practical personal assistant services.

The relationship between training data size and classifier performance was also investigated. We confirmed that the performance becomes better as we use more training data, but the performance increase rate drops as we increase the training data size. All of these observations are reasonable, and we hope that our work can motivate and be referred to by many researchers or industries who are trying to build their own personal assistant systems.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This work was supported by 2017 Hongik University Research Fund, in part by the National Research Foundation of

Korea (NRF) grant funded by the Korea government (Ministry of Science, ICT & Future Planning (MSIP)) (no. 2017R1C1B5015901) and also in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017R1D1A1B03031348).

References

[1] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet computing*, vol. 7, no. 1, pp. 76–80, 2003.

[2] J. B. Greenblatt and S. Saxena, "Autonomous taxis could greatly reduce greenhouse-gas emissions of US light-duty vehicles," *Nature Climate Change*, vol. 5, no. 9, p. 860, 2015.

[3] J. Aron, "How innovative is Apple's new voice assistant, Siri?," *New Scientist*, vol. 212, no. 2836, p. 24, 2011.

[4] D. Sullivan, "Google Now personalizes everyone's search results," *Search Engine Land*, vol. 12, 2009.

[5] A. Warren, *Amazon Echo: The Ultimate Amazon Echo User Guide 2016 Become an Alexa and Echo Expert Now!*, 2016.

[6] D. Yang, D. Zhang, Z. Yu, and Z. Wang, "A sentiment-enhanced personalized location recommendation system," in *Proceedings of the ACM Conference on Hypertext and Social Media*, pp. 119–128, Paris, France, May 2013.

[7] B. Guo, D. Zhang, D. Yang, Z. Yu, and X. Zhou, "Enhancing memory recall via an intelligent social contact management system," *IEEE Transactions on Human-Machine Systems*, vol. 44, no. 1, pp. 78–91, 2014.

[8] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*, MIT Press, vol. 999, MIT Press, Cambridge, MA, USA, 1999.

[9] R. Kuhn and R. De Mori, "The application of semantic classification trees to natural language understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 5, pp. 449–460, 1995.

[10] W. Cui, S. Liu, L. Tan et al., "Textflow: towards better understanding of evolving topics in text," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2412–2421, 2011.

[11] Y. Xu, Z. Ye, L. Chen, S. Li, and G. Pan, "TaskShadow-w: NFC-triggered migration of web browsing across personal devices," in *Proceedings of the ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication (UbiComp'13)*, pp. 99–102, Zurich, Switzerland, September 2013.

[12] L. Chen, G. Pan, and S. Li, "Touch-driven interaction via an NFC-enabled smartphone," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops*, pp. 504–506, Lugano, Switzerland, March 2012.

[13] T. M. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski, "Experience with a learning personal assistant," *Communications of the ACM*, vol. 37, no. 7, pp. 80–91, 1994.

[14] T. Winograd, "Understanding natural language," *Cognitive Psychology*, vol. 3, no. 1, pp. 1–191, 1972.

[15] J. Yi, T. Nasukawa, R. Bunescu, and W. Niblack, "Sentiment analyzer: extracting sentiments about a given topic using natural language processing techniques," in *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pp. 427–434, Melbourne, FL, USA, November 2003.

[16] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[17] W. B. Cavnar and J. M. Trenkle, "N-gram-based text categorization," *Ann Arbor Journal*, vol. 48113, no. 2, pp. 161–175, 1994.

[18] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," *ICML*, vol. 97, pp. 412–420, 1997.

[19] T. Joachims, *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*, Springer, Berlin, Germany, 1998.

[20] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *Journal of Machine Learning Research*, vol. 2, pp. 45–66, 2002.

[21] H. Yu, C. Ho, Y. Juan, and C.-J. Lin, *LibShortText: A Library for Short-Text Classification and Analysis*, Rapport Interne, Department of Computer Science, National Taiwan University, Taipei, Taiwan, 2013.

[22] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: a library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[23] Y. Yang and X. Liu, "A re-examination of text categorization methods," in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 42–49, Berkeley, CA, USA, August 1999.

[24] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1–47, 2002.

[25] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, vol. 10, pp. 79–86, Philadelphia, PA, USA, July 2002.

[26] E. Leopold and J. Kindermann, "Text categorization with support vector machines. How to represent texts in input space?," *Machine Learning*, vol. 46, no. 1–3, pp. 423–444, 2002.

[27] A. Genkin, D. D. Lewis, and D. Madigan, "Large-scale bayesian logistic regression for text categorization," *Technometrics*, vol. 49, no. 3, pp. 291–304, 2007.

[28] M. Lan, C. L. Tan, J. Su, and Y. Lu, "Supervised and traditional term weighting methods for automatic text categorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 721–735, 2009.

[29] T. R. Gruber, A. J. Cheyer, D. Kittlaus, et al., "Intelligent automated assistant," US Patent 9 318 108, 2016.

[30] S. Agarwal, D. Nishar, and A. Rubin, "Providing digital content based on expected user behavior," US Patent 8 271 413, 2012.

[31] S. Agarwal, V. Gundotra, and A. Nicolaou, "Providing results to parameterless search queries," US Patent 8 478 519, 2013.

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[33] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[35] D. Meyer, F. Leisch, and K. Hornik, "The support vector machine under test," *Neurocomputing*, vol. 55, no. 1-2, pp. 169–186, 2003.

[36] V. N. Vapnik and S. Kotz, *Estimation of Dependences Based on Empirical Data*, Springer-Verlag, Vol. 40, Springer-erlag, New York, NY, USA, 1982.
[37] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp. 144–152, Pittsburgh, PA, USA, July 1992.
[38] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
[39] Merriam-Webster.com, "Entry:crowdsourcing," 2012.
[40] M. Buhrmester, T. Kwang, and S. D. Gosling, "Amazon's Mechanical Turk: a new source of inexpensive, yet high-quality, data?," *Perspectives on Psychological Science*, vol. 6, no. 1, pp. 3–5, 2011.
[41] G. Paolacci, J. Chandler, and P. G. Ipeirotis, "Running experiments on Amazon Mechanical Turk," *Judgment and Decision making*, vol. 5, no. 5, pp. 411–419, 2010.
[42] A. Kittur, E. H. Chi, and B. Suh, "Crowdsourcing user studies with Mechanical Turk," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 453–456, Florence, Italy, April 2008.
[43] K.-F. Lee, *Automatic Speech Recognition: the Development of the SPHINX System*, Springer Science & Business Media, vol. 62, Berlin, Germany, 1988.
[44] X. Huang, F. Alleva, H.-W. Hon, M.-Y. Hwang, K.-F. Lee, and R. Rosenfeld, "The SPHINX-II speech recognition system: an overview," *Computer Speech & Language*, vol. 7, no. 2, pp. 137–148, 1993.
[45] W. Walker, P. Lamere, P. Kwok et al., "Sphinx-4: a flexible open source framework for speech recognition," Technical Report, Mountain View, Sun Microsystems Inc., CA, USA, 2004.