Case Study in Developing Extensible Virtual Assistant Using Genie Framework

Yi-Ting Wu[1], Albert Chang[2], Yu Hung Tsai[1], Po-Chuan Wang[1], Tinghao Chen[1] and Jeng-Wei Lin[1][0000-0001-8599-5083]

[1] Tunghai University, Taichung 407224, Taiwan
{g10490001, s08490007, s08490037, g11490031, jwlin}@thu.edu.tw
[2] BSI Pacific, Taiwan Branch, Taipei, 11492, Taiwan
albert.chang@bsigroup.com

Abstract. Deep learning has made significant improvement in natural language processing. Nowadays virtual assistants or chatbots attract attention of many researchers and are expected to be applied in more and more arears. We had designed and implemented an extensible financial virtual assistant using Genie framework. A new device (or skill) is developed to offer financial services in backend server cloud. The device and supported APIs (Application Programming Interface) are registered in an open repository Thingpedia. When Genie receives user utterances, it translates them into ThingTalk programs using a large deep-learning neural networks. Then, Genie executes the ThingTalk programs, which may invoke the financial services through the registered APIs. ThingTalk is a declarative programming language. Domain experts can easily describe financial services in high-level viewpoint with minimal knowledge and experiences of computer programming and system development, while complex services are implemented in backend servers and access through API. As a result, domain experts and computer engineers together can fast and easily build a virtual assistant that support natural language interface.

Keywords: Extensibility, Virtual Assistant, Chatbot, NLP, Large Language Model, Genie, ThingTalk.

1 Introduction

Financial technology, abbreviated as FinTech, is emerging in recent years. Combined together with multiple information and communication technologies (ICT), such as mobile communications, social media, cloud services, and big data analysis, FinTech is expected to significantly change how financial services are provided and consumed. For example, in a user behavior survey of a very popular social app in Taiwan, more than half of its users had accessed official accounts operated by financial services [1, 2]. Chatbots, or virtual assistants, for different financial services become popular in our daily life. However, most of them can understand only common and simple questions and provide answers according to some predefined rules. Using traditional ICT

technologies, it is not easy to create a virtual assistants that can interact with its users in natural languages [3].

Nowadays, various applications have been emerging due to the development of artificial intelligence (AI). Deep learning [4] promotes natural language and speech signal processing significantly. Chatbots and virtual assistants have gained a lot of attention and are applied in many scenarios, such as education, health care, entertainment, and so on [5]. In new scenarios, the dialogue systems of these virtual assistants are improved to make their replies more like human replies [6], and furthermore the way users interact with them is more similar to the way with humans [7].

Virtual assistants, such as Amazon Alexa [8, 9], Google OpenWeave [10], Apple Homekit [11], Samsung SmartThings [12], were designed and developed for these big giant companies. On the other hand, Genie (previously Almond) is an open-source virtual assistant that takes several important issues into consideration, such as privacy, extensibility, and programmability [13-16].

In this paper, we presents an extensible virtual assistant for financial services using Genie framework. Fig. 1 shows the scenario of the proposed virtual assistant. It acts as a chatbot that can provide historical and real-time information of Taiwan Stock market in some chatrooms in a social media. We must note that in practice, there will be user questions that the chatbot cannot understand. It has to handover these questions to the customer support team and technical support team. The latter has to extend the capacity of chatbot while the services cannot stop.

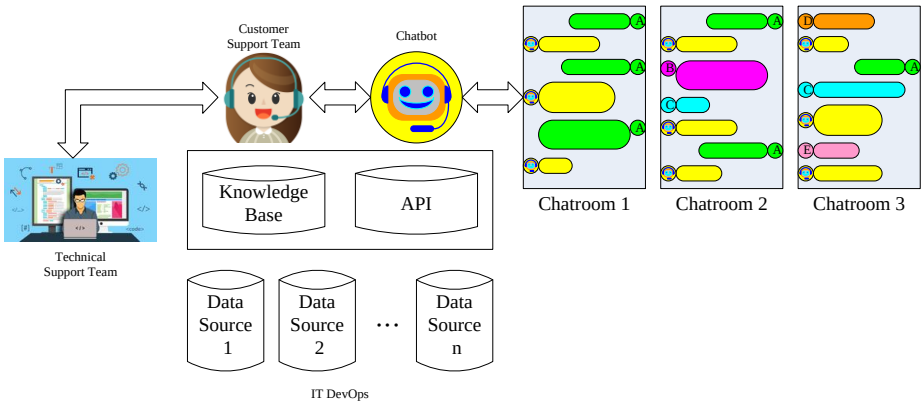

Fig. 1. Scenario of the proposed virtual assistant

We have created the chatbot based on Genie framework [3, 13-16]. A new device (or skill) for Taiwan Stock market is developed to offer stock information in backend server cloud. The device and supported APIs (Application Programming Interface) are registered in an open repository Thingpedia. When Genie receives user utterances, it translates them into ThingTalk programs using a large deep-learning neural networks. Then, Genie executes the ThingTalk programs, which may invoke the financial services through the registered APIs. **Fig. 2** the system architecture of the proposed virtual assistant.

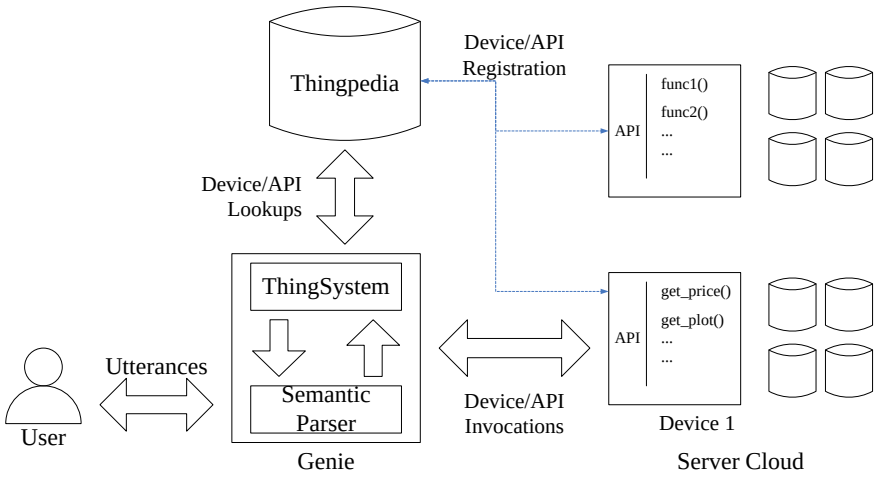

Fig. 2. Proposed virtual assistant architecture based on Genie framework

ThingTalk is a high-level declarative programming language. With a basic level of knowledge and experiences of computer programming and system development, domain experts can conceptually describe financial services in ThingTalk in a high-level viewpoint. Function blocks that together support the financial services physically are implemented by the technical team in backend server cloud. The APIs for these function blocks are registered in Thingpedia. When Genie executes ThingTalk programs, it can look up the requested APIs in Thingpedia and then invoke the corresponding function blocks to realize the financial services. As a result, domain experts and technical team can collaboratively develop and extend charbot fast and easily.

In the remaining of this paper, we will describe the proposed virtual assistant in Session 2, present the preliminary experiment results in Session 3, and give the conclusions in Session 4.

2 Proposed Virtual Assistant

In this session, we first investigate the possible function set of the proposed virtual assistant. As well, we will simply describe ThingTalk programming language. Then, we present the design and implement of the device for Taiwan Stock.

2.1 Function Set of the Virtual Assistant

First, we collected news, reports, press releases, and articles from various platforms for Taiwan Stock. Keywords were identified, such as price, stock names, stock codes, weighted index, and so on. User utterances to query information of these keywords were manually generated, such as the following query utterances.

- Check the trading volume/opening price/highest price/… of today's market?

- Give me the company information of XYZ/ABC/…*?
- Query XYZ/ABC/… stock code?
- XYZ/ABC/… percentage change today
- XYZ/ABC/… daily/weekly/monthly line
- stock (highest, average, and/or lowest) price of XYZ/ABC/… today/yesterday/last week?

*XYZ/ABC/… refer to an abbreviation, full name, or nickname of a stock.

We most note that this collection of user intentions and corresponding utterances is typically incomplete. As we have stated above, there will be always a need to extend the capacity of the proposed chatbot for Taiwan Stock market. Thus, in the beginning, we developed a small set of functions for stock information queries.

2.2 ThingTalk

ThingTalk [13-16] is a high-level declarative language designed to access Internet services and IoT devices. ThingTalk is domain-specific and data focused. It has a very simple construct of three types of clauses: ***stream*** (*s*), ***query*** (*q*), and ***action*** (*a*). The construct follows.

$$s\ [\Rightarrow q]?\ \Rightarrow a; \tag{1}$$

In a ThingTalk program, *s* is a ***stream*** clause that determines when the rest of the program runs. It can be a periodic timer, or it can monitor the result of a ***monitorable query*** function defined in Thingpedia for changes. The optional ***query*** clause (*q*) specifies what data should be retrieved. Results of ***queries*** can be filtered. They can also be used as an input parameter in a subsequent function invocation. The ***action*** clause (*a*) specifies what the program should do.

For example, a user command "notify me when I receive a text" can be done by the following ThingTalk program.

```
monitor (@org.thingpedia.builtin.thingengine.phone.sms())
                        => notify                          (2)
```

To be short, we omit the grammar details of ThingTalk in this paper.

2.3 Device for Taiwan Stock Market

In this study, we registered a new device named as TaiwanStock in Thingpedia, as well as the APIs for the function blocks required to support intended stock services. As shown in **Fig. 3**, a ***query*** function `get_price()` is declared as an API of this device. Again, we omit the detail of the syntax in this paper.

```
Service:TaiwanStock
class @com.TaiwanStock {
    query get_price(
        in opt company : String
          #_[canonical={
            default="base",
            base=["stocknumber", "number","stock code",
                  "stock symbol","ticker symbol"],
            property=["# stock","# stock price"]
          }],
        out stockNo : String
          #_[canonical="stock symbol"],
        out stockName : String
          #_[canonical="stock name"],
    ...)
    ...}
```

Fig. 3. A snapshot of the TaiwanStock device

The three types of clauses in ThingTalk can usually be mapped to three type of phrases in natural languages, and vise versa. For example, a ***stream*** clause can be mapped to ***when*** phrase (***WP***), a query clause to ***noun*** phrase (***NP***), and an ***action*** clause to ***verb*** phrase (***VP***), respectively.

These phrases are used as ***primitive templates*** in genie-toolkit [13-16]. **Table 1** shows some mapping examples used in this study, where genie-toolkit considers ${} as a holder of a parameter. Combined with ***constructive templates*** for the nature language, English in this study, genie-toolkit can generate a large number of user utterances and their corresponding ThingTalk programs. Thus, we can train the semantic parser in Genie to translate user utterances to ThingTalk programs.

Table 1. Some phrases in nature language and ThingTalk clauses.

Phrases	Type of phrases	ThingTalk clauses
price of ${p_code}	***NP***	`@taiwanstock.get_price (company=p_code)`
${p_company}'s K line	***NP***	`@taiwanstock.get_Kplot (company=p_company)`
When ${p_company} rose by more than ${p_change}	***WP***	`edge (monitor (@taiwanstock.get_price (company=p_company))) on change >= p_change`
Call ${p_number}	***VP***	`@org.thingpedia.builtin. thingengine.phone.call (number=p_number)`

3 Preliminary Experiment

3.1 Data Source

In this study, we downloaded the historical data of 2022 Taiwan Stock market from Taiwan Economic Journal (TEJ) [17]. The data was preprocessed and then stored in a SQL database in the backend server cloud.

3.2 Backend Servers

There are two types of backend servers: database servers and application servers. All servers are generic personal computers running Ubuntu 18.04.6 LTS. MariaDB [18] is adopted in the database server. Functions declared in the TaiwanStock device are implemented using Python and Flask [19] according to Restful API design in the application servers.

3.3 Genie Server

Genie server accepts user utterances, translate them into ThingTalk programs by the semantic parsers, and execute the program to fulfill user requests. Genie accesses the TaiwanStock device via the APIs registered in Thingpedia and implemented in the backend application servers.

Genie server is also a generic personal computers running Ubuntu 18.04.6 LTS. It is equipped with AMD® Ryzen 5 2600 six-core processor × 12, 64GB DRAM, and NVIDIA GeForce RTX 2070.

Currently, Genie server adopts BART-base [20] as its semantic parser.

3.4 Experiment

In this preliminary study, we carefully collected 80 user utterances, and designed their corresponding ThingTalk programs. Some user utterances are simple sentences, while others are dialogs of several sentences. **Table 2** shows some of the user utterances and their corresponding ThingTalk programs.

In order to train the semantic parser, BART-base. Phrases were identified as ***WP***, ***NP***, and ***VP***, and mapped to ThingTalk clauses. We used genie-toolkit to generate the training data to train BART-base.

Finally, for the 80 user utterances, we randomly pick up some predicted ThingTalk programs for manual evaluation.

3.5 Experiment Results

It took one hour or so for genie-toolkit to generate the training data and then train semantic parser, currently BART-base.

Table 2. Some user utterances and their corresponding ThingTalk program.

User utterances	ThingTalk program
What is the stock price of 2318?	`$dialogue @org.thingpedia.dialogue.transaction.execute; @com.TWstock.get_price(query="2318");`
Check 2318 stock price on April 15	`$dialogue @org.thingpedia.dialogue.transaction.execute; @com.TWstock.get_price(query="2318", date = new Date("2022-04-15"))`
1101 Trends	`$dialogue @org.thingpedia.dialogue.transaction.execute; @com.TWstock.get_plot(stockname="1101");`

Fig. 4 shows a snapshot when chatting with Genie, where the user requests a K line graph of stock No. 2317.

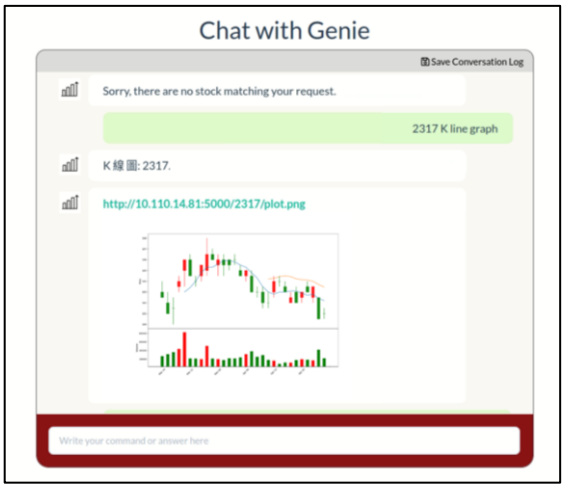

Fig. 4. Snapshot of chatting with Genie

3.6 Evaluation

For the 80 user utterances, we randomly pick up some generated ThingTalk programs for manual evaluation.

Table 3. Some results of the evaluation

No	User utterances	Target ThingTalk program	Predicted ThingTalk program	Evaluation
1	hello	`$dialogue @org.Thingpedia.dialogue. transaction.greet;`	`$dialogue @org.Thingpedia.dialogue. transaction.greet;`	ok
42	tsmc stock price on DATE_0	`$dialogue @org.Thingpedia.dialogue. transaction.execute; @com.TaiwanStock.get_price( date = new Date (2022,3,10, new Time( 8 , 0)), query = " tsmc ") ;`	`$dialogue @org.Thingpedia.dialogue. transaction.execute; [stockName]of @com.TaiwanStock.get_price( date = DATE_0 );`	ok_ function
61	I want to query the stock price of 0050 stock code	`$dialogue @org.Thingpedia.dialogue. transaction.execute; @com.TaiwanStock.get_price( query = " 50 stock code " );`	`$dialogue @org.Thingpedia.dialogue. transaction.execute; @com.TaiwanStock.get_price( query = " 50 " );`	ok_ without_ param
68	query the stock price of 1101	`$dialogue @org.Thingpedia.dialogue. transaction.execute; @com.TaiwanStock.get_price( query=" 1101 " );`	`$dialogue @org.Thingpedia.dialogue. transaction.execute; @com.TaiwanStock.get_price( query = " 1101 " );`	ok

We carefully compare the predicted and target ThingTalk programs [16]. The result of the each comparison could be ok, ok_without_param, ok_function, ok_device, ok_num_function, ok_syntax, or wrong_syntax.

Table 3 show some results of the evaluation. The evaluation result shows 76 of 80 predicted ThingTalk programs are effective.

4 Conclusions and Future Works

In this study, we had designed and implemented an extensible chatbot for Taiwan Stock market based on Genie framework. Genie server accepts user utterances, translate them into ThingTalk programs by the semantic parsers, and execute the program to fulfill user requests. Genie accesses the TaiwanStock device via the APIs registered in Thingpedia and implemented in the backend application servers.

ThingTalk is a declarative programming language. Domain experts can conceptually describe various services in ThingTalk in a high-level viewpoint. They need just a basic level of knowledge and experiences of computer programming and system development. On the other hand, functions actually carrying out the complex computing logics are declared as APIs, and implemented by the technical team in backend server cloud. With ThingTalk programming language as a bridge, domain exports can focus on the service logics in high level, while technical teams can focus on implementing the APIs in the backend servers. As a result, it is much easier to extend the capacity of the chatbot than earlier approaches

4.1 Future works

Currently, several new functions of the chatbot have been identified. New APIs are under construction to extend the capacity of the chatbot. As well, new large language models (LLM), such as ChatGPT [21], are under investigation for translation from user utterances into ThingTalk programs.

References

1. King, B.: Bank 4.0: Banking everywhere, never at a bank. John Wiley & Sons (2018).
2. 2021 LINE User Usage Survey. Nielsen (2021). https://linecorp.com/zh-hant/pr/news/zh-hant/2021/4000, last accessed 2023/5/20. (In Chinese)
3. Wu, Y.-T.: Design and implementation of extensible financial chatbot. Master Thesis. Dept. Information Management, Tunghai University (2023).
4. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521(7553), pp. 436-444 (2015).
5. Liao, S.-w., Hsu, C.-H., Lin, J.-W., Wu, Y.-T., Leu, F.-Y.: A deep learning-based Chinese semantic parser for the Almond virtual assistant. Sensors 22(5), 1891 (2022).
6. Shah, H., Warwick, K., Vallverdú, J., Wu, D.: Can machines talk? Comparison of Eliza with modern dialogue systems. Computers in Human Behavior, 58, pp. 278-295 (2016).
7. Adamopoulou, E., Moussiades, L.: An Overview of chatbot technology. In: Maglogiannis, I., Iliadis, L., Pimenidis, E. (eds) Artificial Intelligence Applications and Innovations.

AIAI 2020. IFIP Advances in Information and Communication Technology, vol. 584. Springer, Cham (2020).
8. Amazon Alexa Voice AI, https://developer.amazon.com/alexa, last accessed 2023/5/20.
9. Goyal, A.; Metallinou, A.; Matsoukas, S. Fast and Scalable Expansion of Natural Language Understanding Functionality for Intelligent Agents. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, LA, USA, 1–6 June 2018.
10. OpenWeave, https://openweave.io/, last accessed 2023/5/20.
11. HomeKit Overview, https://developer.apple.com/apple-home, last accessed 2023/5/20.
12. SmartThings Developers, https://developer.smartthings.com/, last accessed 2023/5/20.
13. Campagna, G., Ramesh, R., Xu, S., Fischer, M., Lam, M. S.: Almond: the architecture of an open, crowdsourced, privacy-preserving, programmable virtual assistant. In Proceedings of the 26th International Conference on World Wide Web, pp. 341-350 (2017).
14. Campagna, G., Xu, S., Moradshahi, M., Socher, R., Lam, M. S.: Genie: A generator of natural language semantic parsers for virtual assistant commands. In Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, 394-410 (2019).
15. Campagna, G., Semnani, S., Kearns, R., Sato, L. J. K., Xu, S., Lam, M.: A few-shot semantic parser for wizard-of-oz dialogues with the precise ThingTalk pepresentation. In Findings of the Association for Computational Linguistics: ACL 2022, pp. 4021-4034. Dublin, Ireland (2022).
16. Genie Wiki, https://wiki.genie.stanford.edu/, last accessed 2023/5/20.
17. Taiwan Economic Journal, https://www.finasia.biz/, last accessed 2023/5/20.
18. MariaDB, https://mariadb.org/, last accessed 2023/5/20.
19. Flask-RESTful, https://flask-restful.readthedocs.io/en/latest/, last accessed 2023/5/20.
20. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. ACL 2020.
21. Introducing ChatGPT, https://openai.com/blog/chatgpt, last accessed 2023/5/20.