Dynamic Service Recommendation Using Lightweight BERT-based Service Embedding in Edge Computing

Author: Kungan Zeng, Incheon Paik

University of AIZU

MCSoC 2021

Outline

- Background

  Service computing for IoT microservices in edge computing.

- Problems

  Dynamic service recommendation, limited computing, and communication resources.

- Method

  BERT-based service embedding

- Experimental Study
- Conclusion

Background

- Service computing refers to service clustering, discovery, recommendation, and composition.
- In this paper, our goal is to perform service recommendation.

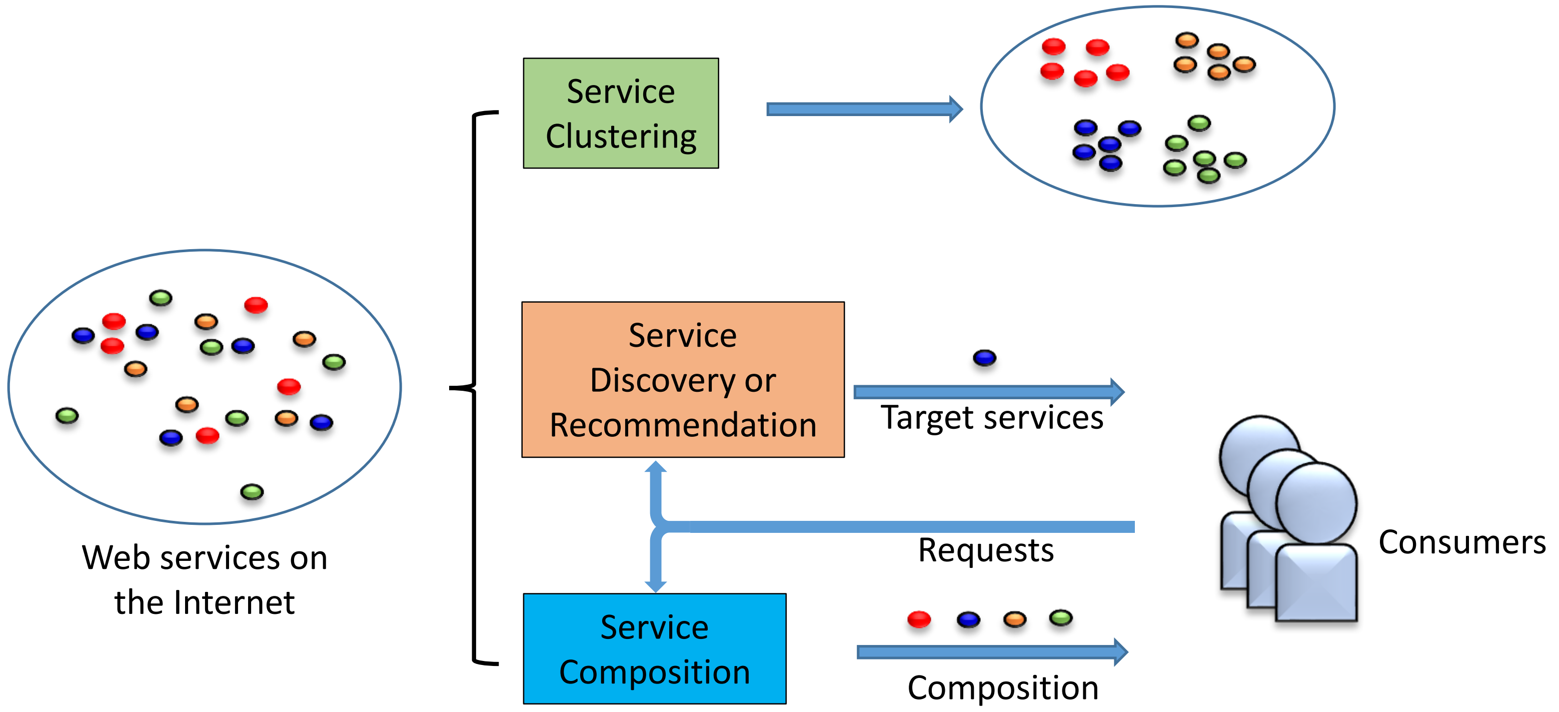


Fig. 1: Service computing

Microservices in IoT systems

- In the distributed environment of IoT systems, a crucial challenge is to build up added-value applications from heterogeneous isolated services.
- Microservices are introduced to separate large, complex, and tightly coupled systems into several microservice component parts for speeding up the development of systems and reducing the cost of making changes.
- As the flexibility, composability, and independence of microservices are very desirable for IoT applications, microservices have been regarded as a potential solution for address the problems in IoT systems.
- To boost the utilization of these microservices, here service recommendation was introduced.

Problems

- Traditional service computing technologies, such as service clustering, discovery, recommendation, and composition, are usually based on centralized environments. Several problems, like the high cost of communication, security, and delay, will emerge when these technologies are applied to distributed environments.

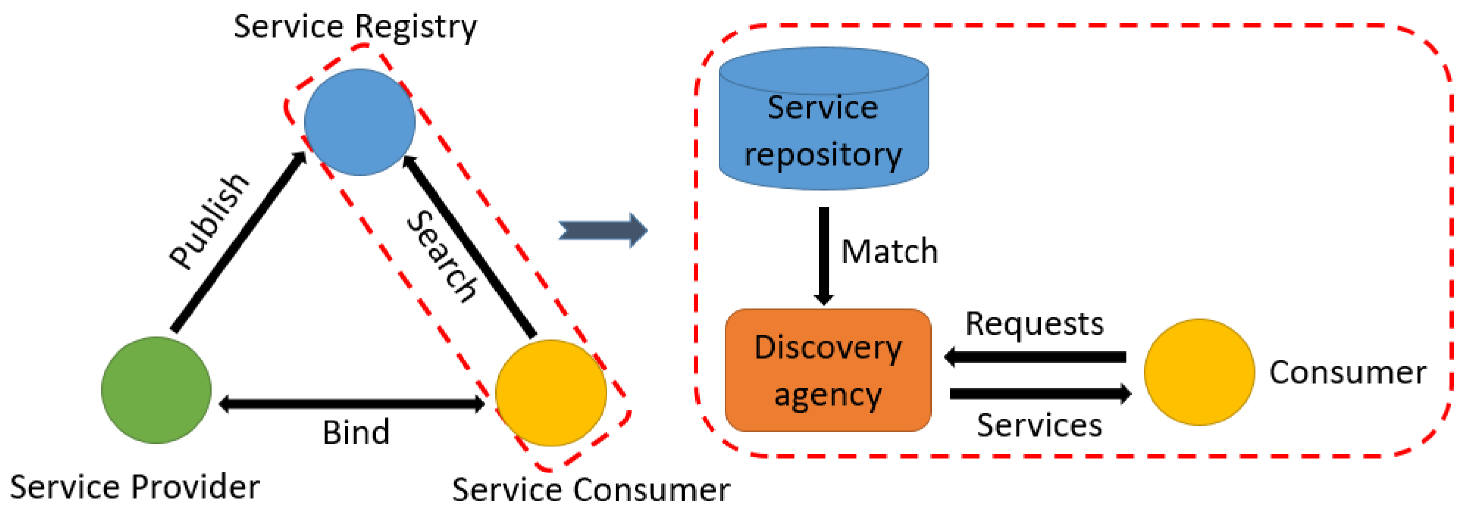


Fig. 2: Web service discovery

Method

Web Service Embedding:

- Service representation using the invocation association between services.
- A deep learning-based method.
- A unsupervised model.

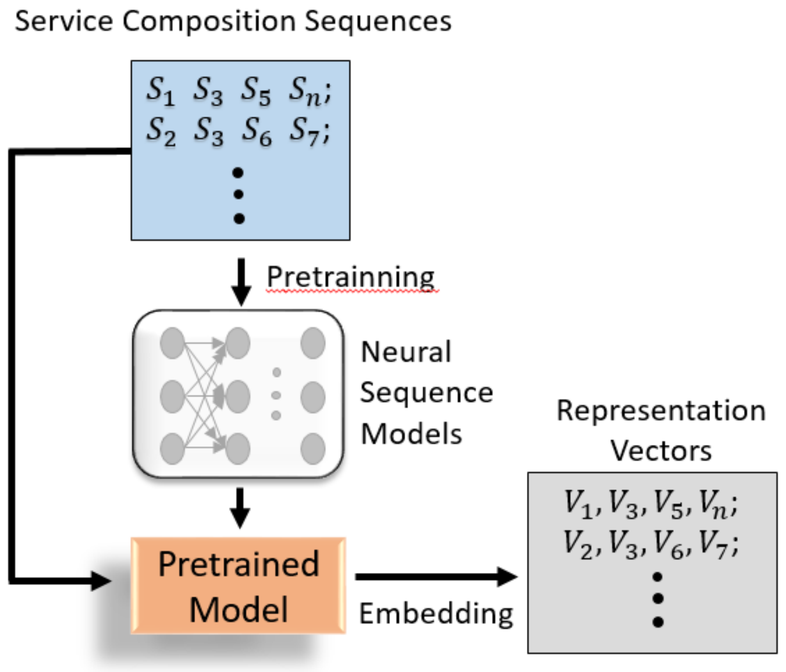


Fig. 3: Web service embedding

BERT

- Bidirectional Encoder Representations from Transformers (BERT) , which has caused a stir in machine learning by demonstrating state-of-the-art results in various NLP tasks, such as machine translation, question-and-answer systems, and chatbots.

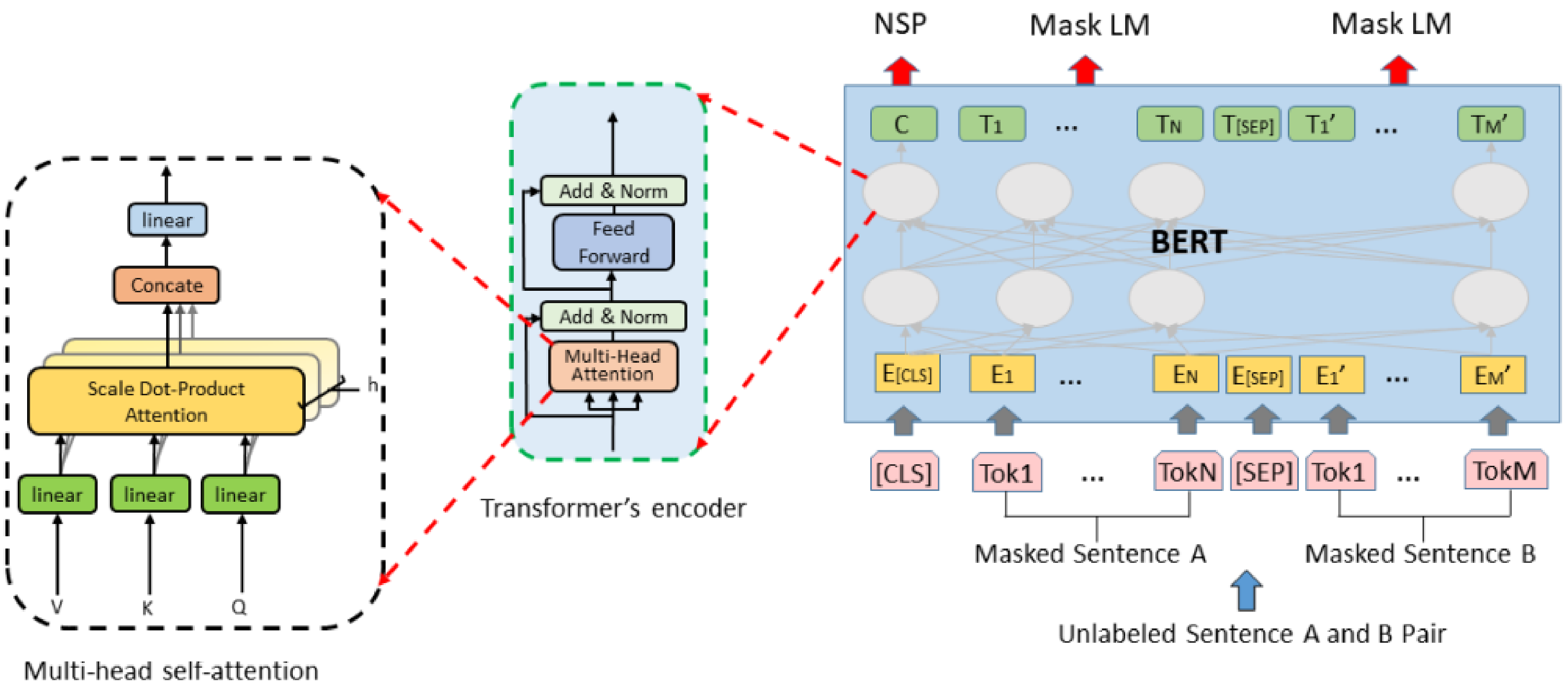


Fig. 4: The architecture of BERT

BERT-based Service Embedding

- Service invocation sequences are considered as token sequences.
- To extract the invocation association between services using the self-attention mechanism.
- To reduce the complexity of the model, a lightweight BERT using convolutional attention was proposed.

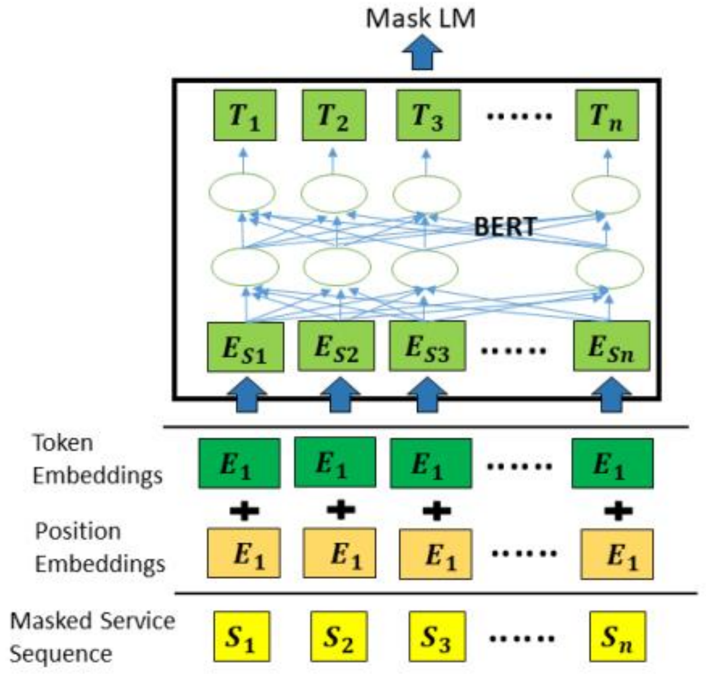

Mask LM
T_1 T_2 T_3 ...... T_n
BERT
E_{S1} E_{S2} E_{S3} ...... E_{Sn}
Token Embeddings
E_1 E_1 E_1 ...... E_1
Position Embeddings
E_1 E_1 E_1 ...... E_1
Masked Service Sequence
S_1 S_2 S_3 ...... S_n

Fig. 5: BERT-based service embedding

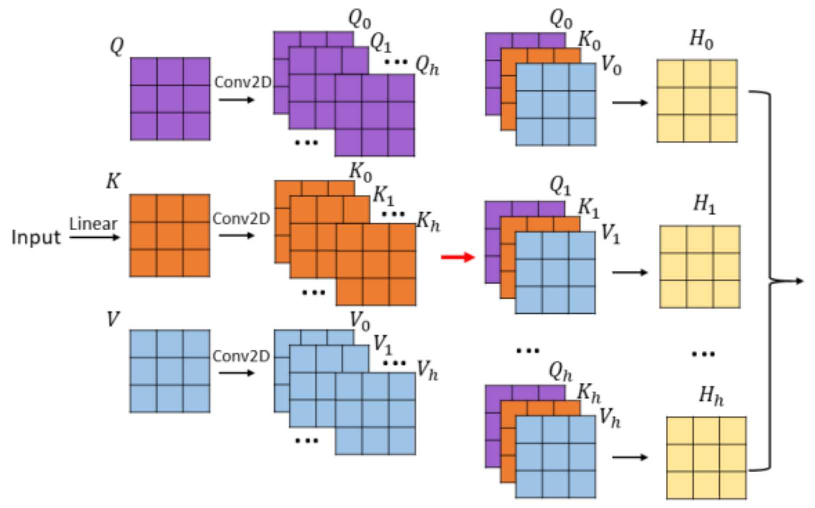


Fig. 6: Convolutional self-attention

Content-based recommendation using service embedding

- Content-based recommendation systems recommend items similar to what the users liked, based on their previous actions or explicit feedback.
- The process consists of two stages, service embedding, and semantic clustering.
- The BERT-based service embedding model is pretrained with service composition sequences, and the semantic information of services can be represented by the embedding process.
- Then, we use K-means clustering, which is an unsupervised method for clustering different semantic categories.

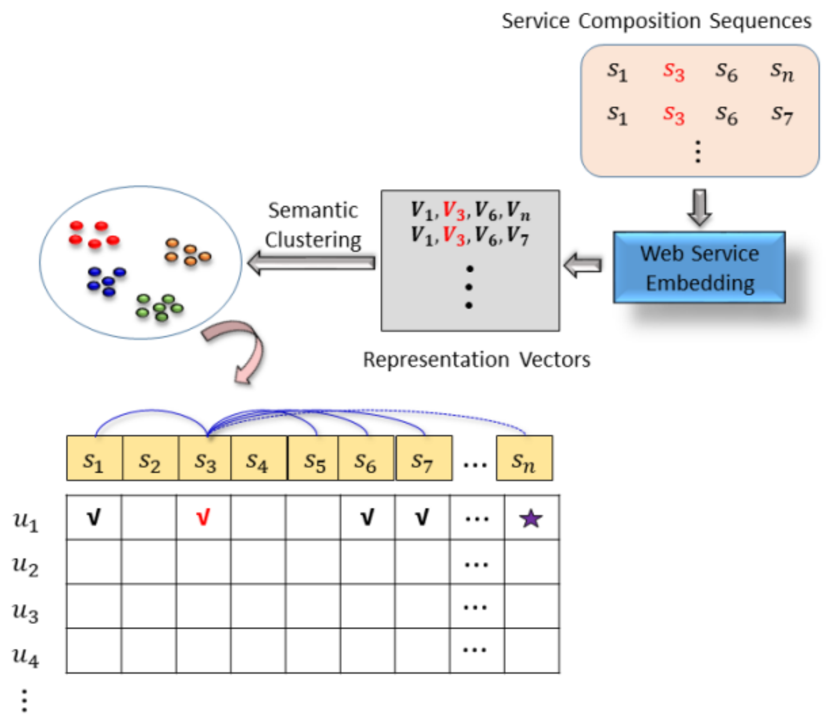


Fig. 7: Service recommendation based on semantic similarity

Dynamic Service Recommendation in Edge Computing

- When our model is implemented in edge nodes, there are four actions, data collection, fine-tuning, recommendation, and data exchange.
- First, the model will be pretrained in the cloud center, then the pretrained model is distributed to each node and utilized to perform service recommendations.
- Second, the model in each edge node collects user historical data and service composition sequences from edge devices and exchanges the data through the cloud center.
- Next, the model starts to perform fine-tuning while the next update cycle coming.
- Finally, service recommendations can continually be performed by the pretrained model, and the whole system can be updated dynamically.

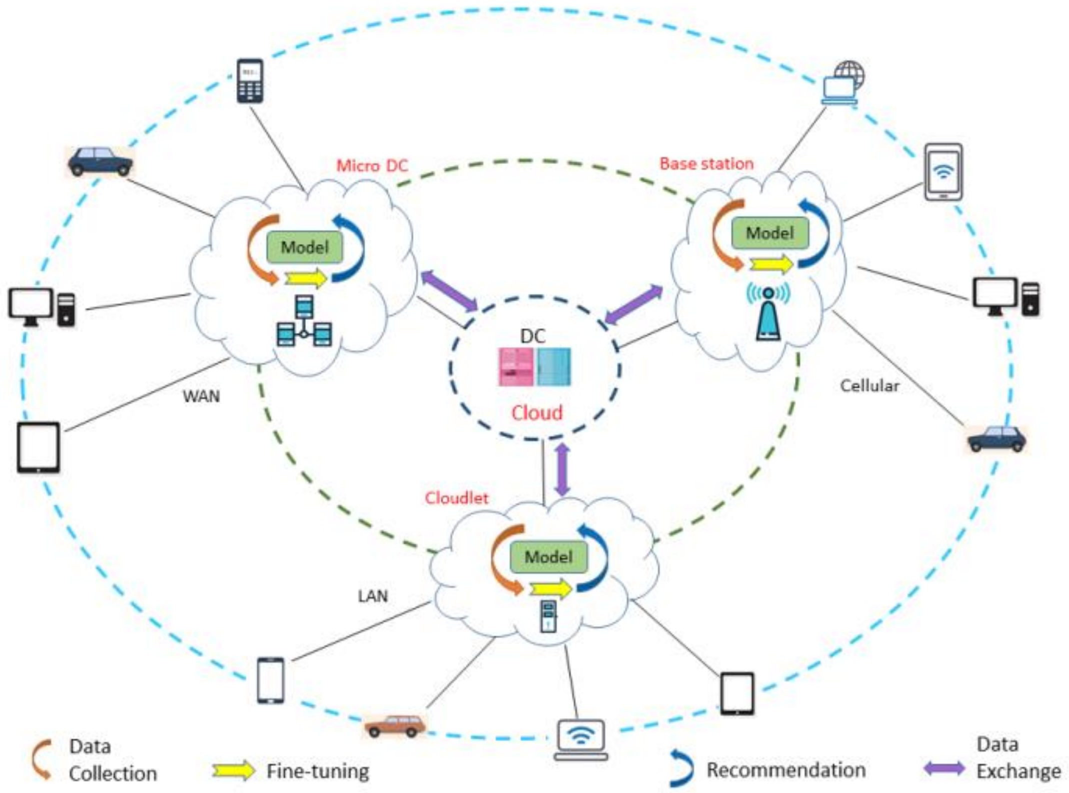


Fig. 7: Dynamic service recommendation in edge computing

Experimental Study

Data preparation:

- First, the source codes are parsed into abstract syntax trees to identify all the methods in each calling method or class.
- Since the research target is the Twitter API, we need to filter some unrelated methods and keep the Twitter API methods only.
- Finally, we can get the Twitter API invocation sequences in a certain definition scope.
- 3000 API invocation sequences, and the total number of methods is about 800.

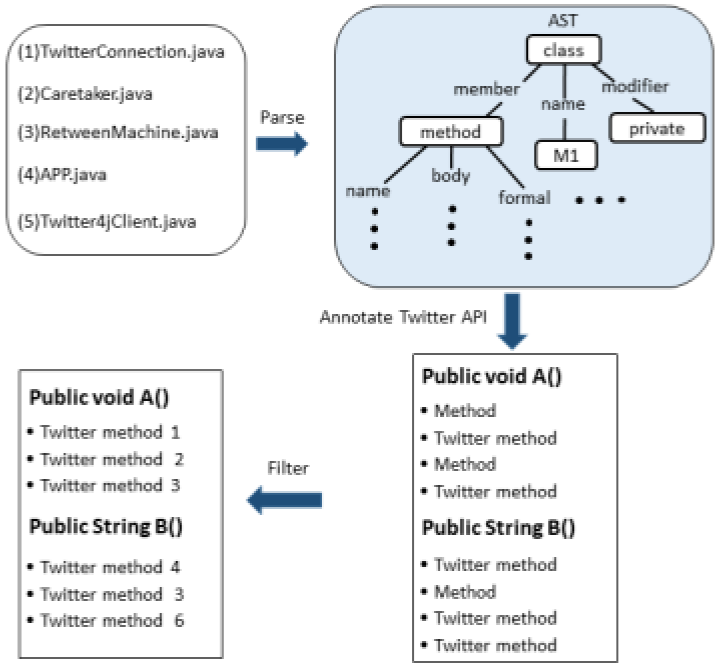


Fig. 8: Data processing

Comparison of the Computational Complexity

- Hyperparameters of the models:

Model	N	d_{model}	d_{ff}	h	**Filter Size**
Base	3	384	768	6	—
lightweight	3	384	768	6	3*9

- Computational complexity:

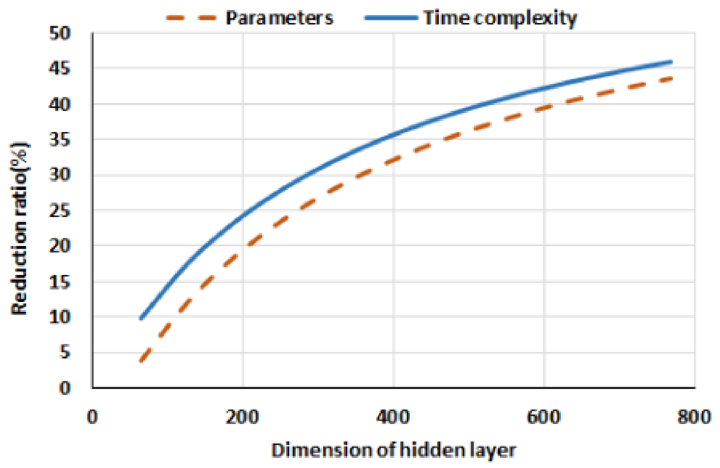
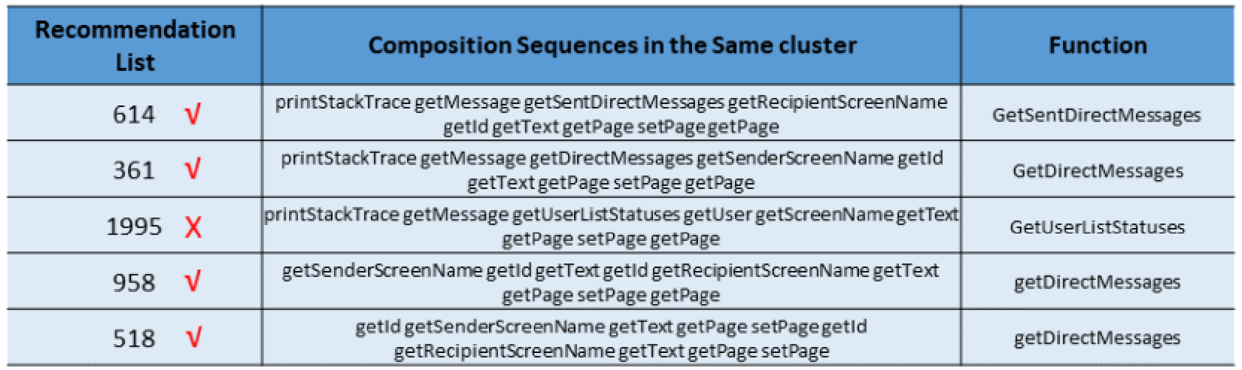


Fig. 8: Reduction ratio of time complexity and parameters

Recommendation Results

- Service sequences with similar functional description will be recommended to users.
- The three largest clusters will be considered for recommendation.



Recommendation List	Composition Sequences in the Same cluster	Function
614 √	printStackTrace getMessage getSentDirectMessages getRecipientScreenName getId getText getPage setPage getPage	GetSentDirectMessages
361 √	printStackTrace getMessage getDirectMessages getSenderScreenName getId getText getPage setPage getPage	GetDirectMessages
1995 X	printStackTrace getMessage getUserListStatuses getUser getScreenName getText getPage setPage getPage	GetUserListStatuses
958 √	getSenderScreenName getId getText getId getRecipientScreenName getText getPage setPage getPage	getDirectMessages
518 √	getId getSenderScreenName getText getPage setPage getId getRecipientScreenName getText getPage setPage	getDirectMessages

Fig. 8: Recommendation examples

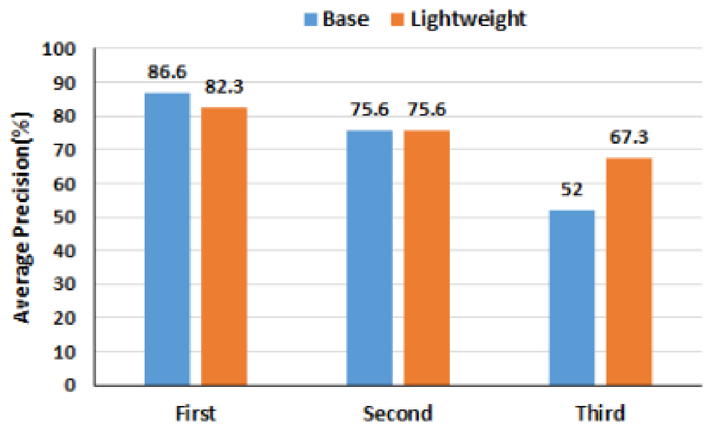


Fig. 8: Recommendation precision

Conclusion

- The experimental results show that our approach performs well for service recommendations.
- At this stage, one limitation of our approach is the recommendation precision is low in small clusters. We will try to improve the performance in future work and extend the experiments in edge computing platforms.