A Selection Support System for Enterprise Resource Planning Package Components using Ensembles of Multiple Models with Round-trip Translation

Masao Ideuchi†,††, Yohei Sakamoto†††, Yoshiaki Oida††††, Isaac Okada††††, Shohei Higashiyama††, Masao Utiyama††, Eiichiro Sumita†† and Taro Watanabe†††

An enterprise resource planning (ERP) package consists of software to support day-to-day business activities and contains multiple components. System engineers combine the most appropriate software components for system integration using ERP packages. Because component selection is a very difficult task, even for experienced system engineers, there is a demand for machine-learning-based systems that support appropriate component selection by reading the text of requirement specifications and predicting suitable components. However, sufficient prediction accuracy has not been achieved thus far as a result of the sparsity and diversity of training data, which consist of specification texts paired with their corresponding components. We implemented round-trip translation at both training and testing times to alleviate the sparsity and diversity problems, adopted pre-trained models to exploit the similarity of text data, and utilized an ensemble of diverse models to take advantage of models for both the original and round-trip translated data. Through experiments with actual project data from ERP system integration, we confirmed that round-trip translation alleviates the problems mentioned above and improves prediction accuracy. As a result, our method achieved sufficient accuracy for practical use.

Key Words: Round-trip Translation, Data Augmentation, Text Classification, Business Data

1 Introduction

An enterprise resource planning (ERP) package consists of software components for constructing an integrated system for managing day-to-day business activities such as finance, human resources, sales, and logistics. To integrate an ERP system for each customer, system designers
must select and combine appropriate components from tens of thousands of choices. In addition to the problem of selecting components from such a large number of candidates, this selection process requires a deep knowledge of business areas and customer requirements. Therefore, only experienced experts can perform this task. For example, to select components for a “Clearing Accounts Payable” operation, a system designer must know its sub-operations, such as “Cash Payment” and “Change Accounting Document.” Additionally, the system designer must be familiar with each business function that is performed by an ERP package’s standard functions, such as “Vendor Line Items” and “Post with Clearing.”

Previous studies have developed machine learning systems that automatically suggest components for ERP system developers (Nakamura et al. 2014; Maniwa et al. 2017; Sakamoto et al. 2018). These systems take functional requirement sentences (FRSs), which are Japanese natural language descriptions referring to customer requirements, as inputs and select ERP package components that correspond to the functionality described in the FRSs. However, these systems face two major difficulties caused by the specification of components in FRSs. The first problem is data sparsity, meaning that insufficient data exacerbate the zero-frequency problem. The second problem is the expression diversity of FRSs, which means that each customer may use different expressions to refer to identical components.

We cast the task of suggesting ERP components as a classification task in which a set of suitable components is predicted based on given input descriptions. To solve the problems of data sparsity and expression diversity, we propose a method that uses ensembles of multiple models with neural machine translation (NMT) round-trip translation. The proposed method consists of three steps. 1) Create augmented training and testing data through the round-trip translation of training and testing data, respectively, using domain-adapted N-best translation. 2) Create multiple models that are trained using either the original data or augmented data. 3) Use ensembles of prediction results from each model: one from the original testing data predicted by a model trained on the original training data and the other from the augmented testing data predicted by a model trained on the augmented training data.

We performed experiments using practical data constructed from real documents on past projects. The experimental results demonstrate that the proposed techniques improve accuracy and the final system achieved practical performance. Our main findings can be summarized as follows:

• An ensemble method that combines models trained separately on original data and augmented data performs better than a single model trained on a mixture of both types of data.
• The domain adaptation of translation models and N-best translation improve the quality of augmented data.
• Our ensemble method with round-trip translation is applicable to different neural architectures, including models based on both long short-term memory (LSTM) and bidirectional encoder representations from transformers (BERT).

Our error analysis reveals that the domain-adapted NMT generates diverse expressions for a single component, which may increase generalization ability for unseen inputs. Subjective analysis revealed that our method achieves accuracy levels that can satisfy the requirements for practical use.

2 ERP Package Component Selection

2.1 Background

ERP packages are very flexible in their ability to combine various functions and components, and it is simple for a system designer to construct a system at a low cost using such packages. Based on these features, ERP packages have been widely used since the 1990s for the purposes of standardization and system cooperation (Mabert et al. 2000; Olhager and Selldin 2003). The number of components has also increased with the introduction of new technologies such as automation and prediction (Madakam et al. 2020). Therefore, the number of components has become increasingly large, reaching as high as tens of thousands of enterprise system functions provided by a single ERP package. Furthermore, the detailed functionality of each component can be changed by changing parameter settings. Therefore, no individual engineers can understand all components in detail.

When constructing a system, system designers must select components that satisfy customer requirements for every project because requirements differ from customer to customer. Expert system designers can select and set parameters effectively based on their knowledge of the business area and past projects. However, non-experts, even if they are expert software engineers, cannot narrow the scope of investigation to select appropriate components. The incorrect selection of components leads to additional implementation work and inferior system quality, as well as increased cost and delivery time. System designers require many years of training to work as experts in this field based on the need to acquire knowledge and experience from many projects. Therefore, the number of experts cannot increase rapidly.

Selecting components is an important part of a project when designing an ERP system and it tends to be a bottleneck. ERP package component selection support systems are designed
to support designers by automatically suggesting appropriate components for a project. Once a candidate component is selected, it is trivial for standard software engineers, even non-experts, to verify whether the selected component satisfies customer requirements by actually operating the component.

To support software development, various recommendation systems for software libraries or APIs have been developed (Zhou and Resnick 2009; Ouni et al. 2017; Almarimi et al. 2019; Chen and Xing 2016). Zhou and Resnick (2009) found that conversation co-mentions and text matching from Drupal.org online forums are effective for software module recommendation based on collaborative filtering. In particular, co-mentions work well with popular modules and text matching works well with less popular modules. Ouni et al. (2017) and Almarimi et al. (2019) used software descriptions as a data source to calculate the similarity between third party libraries. Chen and Xing (2016) used tags for online articles and constructed a knowledge base to find analogical libraries in other programming languages. The main targets of previous studies on software development support are open-source software, so the usage histories and descriptions of software can be easily obtained. In contrast, the usage histories and descriptions of ERP components are difficult to obtain because many ERP components are used to construct proprietary systems and the information of such systems is kept secret. Therefore, to support software development using ERP components, it is necessary to use limited resources effectively.

In a previous study on ERP software development support, Nakamura et al. (2014) used a business function ontology to suggest related components and graphically display the relationships between components as an explanation for suggestions. They combined business domain knowledge and the text similarity of ERP component descriptions and customer requirements, and achieved a high recall because their business function ontology covered related functions, but the precision was low because all components in the same category were not necessary for every project. Sakamoto et al. (2018) simplified ERP software development support as a classification task that uses ERP component text descriptions and introduced round-trip translation to increase the amount and variation of FRSs. Round-trip translation is a method for augmenting data by translating text from one language to another language, and then translating backward. This is a well-known data augmentation technique in the field of machine translation. Their work demonstrated that data augmentation and ensemble prediction yield improved accuracy, although the final accuracy still did not satisfy the requirements for practical use. In this study, we developed a system that performs a classification task, similar to Sakamoto et al. (2018), to suggest ERP components.
2.2 Approaches to Data Sparsity and Diversity Problems

There are three well-known approaches to improving the accuracy of text classification: data augmentation, using a pre-trained model with fine-tuning, and using ensembles of prediction results. To solve the problem of data sparsity and diversity, data augmentation increases the amount and diversity of data, and the use of a pre-trained model exploits the semantic similarity of text from a large amount of pre-trained data. Finally, using ensembles of prediction results enhances the generalization ability of models.

Data augmentation methods for text data include the use of round-trip translation and word replacement (Liu et al. 2020). Most existing data augmentation methods (Lu et al. 2006; Ragni et al. 2014; Ohga et al. 2018) increase the accuracy of learning models on general public data such as conversation data. However, we focus on how data augmentation contributes to increasing the accuracy of business data.

Round-trip translation increases the amount of data by obtaining paraphrases using machine translation (Yu et al. 2018; Wieting et al. 2017; Wieting and Gimpel 2018; Hu et al. 2019a, 2019b). Recently, it has become easier to perform NMT locally because NMT libraries (Ott et al. 2019) and parallel corpora (Morishita et al. 2020) can be obtained easily. Therefore, applying NMT round-trip translation to business data has also become easier because local NMT maintains data confidentiality. Furthermore, we can expect to obtain high-quality translations of target domain text through the domain adaptation of NMT (Koehn and Knowles 2017; Chu and Wang 2018), which leverages an in-domain corpus in addition to a general domain corpus.

Word replacement generates similar text data by either substituting a word in text (Kobayashi 2018; Wei and Zou 2019), or by swapping or deleting words (Wei and Zou 2019). Wei and Zou (2019) introduced easy data augmentation (EDA), which improves text classification accuracy and consists of four simple operations. Two of the operations use synonyms to replace or insert words in the text data. The other operations are the deletion and swapping of words.

Pre-trained language representation models, which are trained on large amounts of text, have been shown to improve various tasks in natural language processing (NLP). The transformer model (Vaswani et al. 2017) was recently proposed and has become the state-of-the-art method in NMT. Additionally, pre-trained models such as BERT (Devlin et al. 2019), XLNet (Yang et al. 2019), RoBERTa (Liu et al. 2019), ALBERT (Lan et al. 2020), and ELECTRA (Clark et al. 2020) are used in various NLP tasks. For handling Japanese, there are some BERT models available that are pre-trained on Japanese text (Shibata et al. 2019; Suzuki 2019; National Institute of Information and Communications Technology 2020).

Constructing ensembles of multiple models yields greater generalization ability than using...
each model individually (Sagi and Rokach 2018). Different ensemble methods vary in complexity. Simpler methods calculate the linear sum of model results (Breiman 1996; Wolpert 1992). The linear sum is analogous to the “or” operation and works well for integrating different predictions. The product of experts (PoE) (Hinton 2002) multiplies similar probability distributions from multiple models. PoE is analogous to the “and” operation and works well for integrating similar predictions. More complex methods use deep learning in ensemble models (Coscrato et al. 2020).

3 EMMRT: Ensemble of Multiple Models with Round-trip Translation

3.1 Task Definition of ERP Package Component Selection Support

For an ERP package component selection support system, we considered component selection as a classification task. The data used for training were labeled FRSs from past projects and the data for prediction were FRSs without labels from new projects. Each labeled FRS is a sample \((x, y)\), where \(x\) is an FRS that specifies a component with the necessary function, \(y \in L\) is a component ID, which is treated as a class label, and \(L\) is the set of all labels. The task requires the return of \((x, y^*)\) from an input \(x\), where \(x\) is an FRS from a new project and \(y^*\) is the predicted label.

To develop an ERP package component selection support system, we divided the labeled data \(D\) into training data \(D_{\text{train}}\) and testing data \(D_{\text{test}}\). The prediction for \(x\) is defined as

\[
y^* = \text{Predict(Model}(M, D), x),
\]

where Model\((M, D)\) is the model trained for \(D\) using method \(M\). Similarly, the predictions \(Y^*\) for a set of FRSs \(X = \{x_1, ..., x_n\}\) are denoted as

\[
Y^* = \text{Predict(Model}(M, D), X)
= \{\text{Predict(Model}(M, D), x_1), ..., \text{Predict(Model}(M, D), x_n)\}.
\]

The ensemble result of predictions from two models for \(x\) is defined as

\[
y^* = \text{Ensemble}(E, \{\text{Predict(Model}(M_1, D_1), x), \text{Predict(Model}(M_2, D_2), x)\}),
\]

where \(E\) denotes the ensemble method. For data augmentation, we denote the augmented data as \(\text{Aug}(D)\), where \(\text{Aug}\) represents augmentation and is replaced by the name of the augmentation method. For example, if the augmentation method is round-trip translation (RT in the equation)
and the LSTM model is trained using augmented data combined with original data, we define the prediction for $x$ as

$$y^* = \text{Predict(Model(LSTM, \{D_{train} \cup RT(D_{train})\}), x).}$$

3.2 Overview of EMMRT: Ensemble of Multiple Models with Round-trip Translation

We propose a method called EMMRT to improve classification accuracy based on text data. EMMRT uses ensembles of multiple models, where each model is trained either on the original data or on data augmented by round-trip translation. An overview of EMMRT is presented in Fig. 1.

Fig. 1 Overview of EMMRT
EMMRT has the following features.

1. It uses pre-trained BERT for computing contextualized embedding and a fully connected linear network that minimizes cross-entropy loss for multi-class classification.
2. Each of the two classifiers is fine-tuned for FRSs using the augmented training data from round-trip translation based on domain-adapted translation models.
3. It uses ensembles of separately trained models: one trained on the original training data and the other trained on the augmented training data.

First, we prepared an NMT system trained on a large general bilingual corpus and a small ERP-related bilingual corpus, and then obtained the round-trip translation data for the training and testing data. Next, we separately trained models on the original training data and augmented training data. Finally, we applied the ensemble principle to the prediction scores of the models. One score was obtained from the original testing data and predicted by the model that was trained on the original training data. The other score was the average score from the augmented testing datasets that was predicted by the model that was trained on the augmented training data. We defined a hyperparameter to determine the best mixing ratio based on a linear sum of prediction scores.

EMMRT can be defined as follows according to our notation:

\[
\text{Ensemble}(\text{Linear}(a:b), \{\text{Predict(Model}(\text{jBERT}, D_{\text{train}}), D_{\text{test}}),\}
\]

\[
\text{Predict(Model}(\text{jBERT}, \text{adaptRT}_n(D_{\text{train}})), \text{adaptRT}_n(D_{\text{test}}))\}),
\]

where Linear\((a:b)\) denotes a linear sum of the prediction scores with a ratio of \(a:b\), \(\text{jBERT}\) denotes BERT trained on Japanese text, and \(\text{adaptRT}_n(D)\) denotes the round-trip translation result of \(D\) from the domain-adapted NMT using N-best translation. The mixture of the original data and round-trip translated data is denoted as \(\{D \cup \text{adaptRT}_n(D)\}\).

### 3.3 Data Augmentation for Training Data

Improving the accuracy of prediction requires a large amount and diversity of training data. In other words, the prediction of component IDs requires more labeled FRSs as training data. However, it is difficult to prepare a large amount of business data. To increase the number of labeled FRSs, we used ja-en and en-ja round-trip translation to obtain FRSs with similar meanings, but different expressions. By using N-best translation, we controlled the number of FRSs in the augmented training data. The number of FRSs became \(N \times M\) times larger when N-best translation for ja-en and M-best translation for en-ja were performed.

As mentioned previously, \(D_{\text{train}}\) denotes the original training data and \(\text{adaptRT}_n(D_{\text{train}})\)
denotes the training data augmented by the domain-adapted NMT using N-best translation. For comparisons to a baseline method, we also used RT($D_{\text{train}}$) and RT$_n$($D_{\text{train}}$), which are outputs from round-trip translation without domain adaptation. The character $n$ following RT indicates that N×M-best round-trip translation is used for augmentation.

### 3.4 Data Augmentation for Testing Data

In EMMRT, data augmentation is applied to both the training data and testing data, which are round-trip translated and fed into the model trained on round-trip translated data. This means that round-trip translation is also executed in the prediction phase. The final prediction is the ensemble result of two predictions: one from the FRSs in a new project and the other from the round-trip translation results of the FRSs. Because the NMT model is trained on both an ERP-related corpus and general corpus, the round-trip translated FRSs are expected to contain more diverse expressions than the original FRSs. Because we apply round-trip translation to both the training data and testing data, we can narrow the gap between the expressions in the training data and those in the testing data. N×M-best round-trip translation is also useful for testing data augmentation because translation candidates further reduce this gap.

Similar to the training data, we denote the original testing data as $D_{\text{test}}$ and augmented testing data (augmented by domain-adapted NMT using N-best translation) as adaptRT$_n$($D_{\text{test}}$). RT($D_{\text{test}}$) and RT$_n$($D_{\text{test}}$) are the outputs of round-trip translation without domain adaptation.

### 4 Experiments

In this section, we demonstrate that each element of EMMRT contributes to improving the accuracy of the LSTM and BERT methods by comparing it to a baseline method and the results of an ablation study. The experimental settings are explained first, followed by the experimental results.

#### 4.1 Experimental Settings

##### 4.1.1 Data Preparation

We collected data from actual ERP package system integration data and explanatory descriptions of standardized business activities published on a website by SAP\footnote{https://www.sap-tcodes.org/} and METI\footnote{https://www.meti.go.jp/policy/servicepolicy/contents/management_support/files/download.html}. These data consist of FRSs and their corresponding components. Each component has a component ID,
Ideuchi et al. A Selection Support System for ERP Package Components using EMMRT

which is an identifier of that component and is attached to each FRS as a class label. Examples of the training data are presented in Table 1. We gathered FRSs from three types of documents: 765 FRSs from requirement documents for real projects, 3,318 FRSs from explanations of components written in operational workflow and incident logs of specific components, and 1,067 FRSs from public information provided by SAP and METI.

The total number of classes in the training data was 521 and multiple labels were assigned to many instances. However, more than 85% of the instances had only a single label, so we simplified the task to single-label classification by expanding a multi-label instance into multiple single-label instances, where all labels were correct. For example, a training instance \( x \) with two labels \( y_1 \) and \( y_2 \) was treated as two instances \((x, y_1)\) and \((x, y_2)\), and both expanded instances were assessed as correct for loss calculation if either \( y_1 \) or \( y_2 \) was predicted. There were 314 classes with one to five instances, 107 classes with 6 to 10 instances, and 100 classes with 11 or more instances. Therefore, more than half of the classes contained five or fewer instances and there was significant imbalance between classes. To alleviate this imbalanced distribution, we increased the number of instances of minority classes by applying \( N \times M \)-best round-trip translation to the training data according to the number of instances in the original training data to obtain more than 10 instances of each class after round-trip translation. For example, a \( 4 \times 3 \)-best round-trip translation was applied to a class that had only one instance in the training data and a \( 3 \times 2 \)-best round-trip translation was applied to a class that had two instances in the training data. We used GiNZA\(^3\) (Matsuda 2020) as a tokenizer for LSTM-based models.

The testing data consisted of FRSs extracted from the same source documents as the training data under the condition that the components had two or more instances. A total of 506 instances were included. We applied a \( 2 \times 2 \)-best round-trip translation to the testing data and calculated an average score from four testing sets.

| Component ID (label) | FRS (in Japanese) | |
|----------------------|------------------|---|
| F110                 | 買掛債務管理 購買業務 決済 支払実行 - 支払を実行する | |
| AS01                 | 固定資産管理 資産取得 資産取得申請 - - 固定資産取得の申請を行う | |
| FB01                 | 単体決算業務 決算手続 決算手続 各引当金計上 - 各引当金を計上する | |

Table 1 Examples of training data

\(^3\)https://megagonlabs.github.io/ginza/

Our preliminary experiments revealed that GiNZA performed better than Janome. Therefore, we employed Ginza for the entire study using the LSTM-based model reported in this paper.
To prepare NMT systems, we used the JParaCrawl Parallel corpus (v2.0)\(^4\) as a general corpus for training and an approximately 40,000 word in-house ERP parallel corpus (collected from different sources of FRSs) for domain adaptation.

### 4.1.2 NMT Systems for Round-trip Translation

We prepared two NMT systems for our experiments. One was a general NMT system that was trained on a general corpus and the other was a domain-adapted NMT system that was trained on a general corpus and an ERP corpus. We selected fairseq\(^5\) as a modeling toolkit and SentencePiece\(^6\) (Kudo and Richardson 2018) as a sentence tokenizer. We split sentences with a vocabulary size of 32,000 and set the learning parameters based on the WMT19 NMT robustness task (Berard et al. 2019; Ott et al. 2018; Vaswani et al. 2017). We then removed sentences with lengths exceeding 250 sub-words. The validation and testing sets, both of which contained 10,000 sentences, were extracted from the complete data of JParaCrawl. After training the translation model for 50 epochs on approximately 10 million sentences, the BLEU scores (Papineni et al. 2002) on the testing set were 29.4 for ja-en and 30.9 for en-ja.

We also investigated the effects of NMT domain adaptation. First, we split the ERP corpus into a training set (99.0%), validation set (0.5%), and testing set (0.5%). We oversampled (by 250 times) the ERP training set and merged it with the JParaCrawl training set before training the domain-adapted translation model. This is because some previous studies (Morishita et al. 2020; Song et al. 2020) reported that oversampling is better than fine-tuning for domain adaptation. The BLEU scores for the testing set of the ERP corpus increased to 46.8 for ja-en and 34.6 for en-ja. These scores are 15.2 and 8.2 points higher, respectively, than those of the model trained using only JParaCrawl.

### 4.1.3 Classification using BERT Models

We used the simple transformers\(^7\) implementation of a BERT-based classifier. In this model, the BERT output layer is followed by a linear layer that transforms an input vector into another vector whose dimension is equal to the number of classes, as shown in Fig. 1. Japanese text preprocessing for each BERT model was performed based on the experimental procedure used in the NICT BERT.\(^8\)

\(^4\) [http://www.kecl.ntt.co.jp/icl/lirg/jparacrawl/](http://www.kecl.ntt.co.jp/icl/lirg/jparacrawl/)
\(^5\) [https://github.com/pytorch/fairseq](https://github.com/pytorch/fairseq)
\(^6\) [https://github.com/google/sentencepiece](https://github.com/google/sentencepiece)
\(^7\) [https://simpletransformers.ai/](https://simpletransformers.ai/)
\(^8\) [https://alaginrc.nict.go.jp/nict-bert/Experiments_on_RCQA.html](https://alaginrc.nict.go.jp/nict-bert/Experiments_on_RCQA.html)
First, we fine-tuned the BERT models using the original data. All BERT models produce outputs with poor accuracy after one epoch, so we changed the number of fine-tuning epochs to 50. To compare accuracies more precisely, because the fine-tuned BERT base model sometimes produced better results than the fine-tuned BERT large model using all training data, we divided the training data into multiple training and validation sets using a stratified 10-fold split (Kohavi 1995) and calculated the average accuracy of experiments using the BERT models. We then calculated the prediction scores of multiple models that were fine-tuned either on the original data or augmented data for the ensemble based on the testing data. In this paper, we report the results of the Kyoto large BERT model.9

4.1.4 Evaluation Metrics

We adopted an accuracy indicator that can evaluate ranking accuracy because it outputs the ranking of the predicted component IDs ordered by certainty. Specifically, the results are evaluated by the P@k indicator, which is defined as the proportion of results in which the correct component ID is within the top-k-ranked component IDs predicted by the proposed system. We used P@1, P@3, and P@20 for comparisons to the baseline method and from the perspectives of whether the predictions of EMMRT are correct, whether the correct predictions lie near the top, and whether system designers can find correct components by examining prediction results. The target prediction accuracy for EMMRT is P@3=80% (80% within the top three) because poor prediction accuracy would waste subsequent work. We also calculated the mean reciprocal rank (MRR) as a total ranking score.

4.2 Main Results

We selected the method from Sakamoto et al. (2018) as a baseline method and refer to their method as the mixed training method with round-trip translation (MTRT). MTRT uses a unidirectional LSTM followed by a linear layer with softmax activation as a classification model and Janome10 as a tokenizer. Notably, MTRT extracts only nouns from original FRSs and uses noun sequences as inputs for its classification model. It merges round-trip-translated data using general NMT with original data for training and separates the round-trip-translated data from the original data for testing. To calculate results, it combines the scores of each component ID

---

9 https://nlp.ist.i.kyoto-u.ac.jp/index.php?ku_bert_japanese
We examined BERT models that were pretrained on Japanese or multiple languages from Google, Kyoto University, Tohoku University, and NICT in our preliminary experiments. The Kyoto large BERT model performed the best in most experiments.

10 https://mocobeta.github.io/janome/en/
from the original testing data and round-trip-translated testing data with a ratio of 2:1. The formulation for their method is defined as follows:

\[
\text{Ensemble}(\text{Linear}(2:1), \{\text{Predict(Model(LSTM, \{D_{train} \cup RT(D_{train})\}), D_{test}),}
\]

\[
\text{Predict(Model(LSTM, \{D_{train} \cup RT(D_{train})\}), RT(D_{test}))},
\]

where \(D_{train} \cup RT(D_{train})\) represents the mixture of the original training data and round-trip-translated training data. The baseline method constructs one model from the mixed training data and combines the prediction scores from the original testing data and the round-trip-translated testing data.

We also compared our results to those of three other methods. The first was fine-tuned BERT in the form of two fine-tuned pre-trained BERT models using the original training data and an ensemble of two models with a mixing ratio of 0.5 for fair comparison because EMMRT uses two BERT models. The second was EDA as an alternative data augmentation method. And the third was PoE as an alternative ensemble method. For comparisons to the baseline method, we trained each model for 50 epochs on all training data. The results of comparisons in terms of classification accuracy are presented in Table 2. For most experiments, the table contains two rows that represent the linear sum ensemble source scores from the different sets of training data and testing data described in the first and second rows. The third column presents the ratio of Linear(1st:2nd). The results of the EDA experiment are presented in a single row because it is

| Experiment setting | method | 1st:2nd ratio | ID | Training data | Test data | P@1 | P@3 | P@20 | MRR |
|--------------------|--------|---------------|----|--------------|-----------|-----|-----|------|-----|
| MTRT (baseline)    | LSTM   | 2:1           | 1st | \(D_{train} \cup RT(D_{train})\) | \(D_{test}\) | 54.94 | 69.56 | 86.36 | -   |
|                    |        |               | 2nd | \(D_{train} \cup RT(D_{train})\) | RT(\(D_{test}\)) |     |     |      |     |
| EMMRT (ours)       | jBERT  | 8:2           | 1st | \(D_{train}\) | \(D_{test}\) | 68.58 | 83.40 | 90.71 | 0.763 |
|                    |        |               | 2nd | adaptRTn(\(D_{train}\)) | adaptRTn(\(D_{test}\)) |     |     |      |     |
| Fine-tuned BERT    | jBERT  | 5:5           | 1st | \(D_{train}\) | \(D_{test}\) | 67.98 | 82.41 | 91.90 | 0.761 |
|                    |        |               | 2nd | \(D_{train}\) | \(D_{test}\) |     |     |      |     |
| Fine-tuned BERT + EDA | jBERT | -             | - | \(D_{train} \cup EDA(D_{train})\) | \(D_{test}\) | 66.01 | 79.84 | 88.34 | 0.737 |
| EMMRT (ours) + PoE for ensemble | A jBERT | 8:2 | 1st | \(D_{train}\) | \(D_{test}\) | 68.58 | 82.81 | 91.11 | 0.763 |
|                    |        |               | 2nd | adaptRTn(\(D_{train}\)) | adaptRTn(\(D_{test}\)) |     |     |      |     |
|                    | B jBERT | 8:2 | 1st | \(D_{train}\) | \(D_{test}\) | 68.58 | 82.81 | 91.11 | 0.763 |
|                    |        |               | 2nd | RTn(\(D_{train}\)) | RTn(\(D_{test}\)) |     |     |      |     |
|                    | C jBERT | 8:2 | 1st | \(D_{train}\) | \(D_{test}\) | 68.58 | 82.81 | 91.11 | 0.763 |
|                    |        |               | 2nd | adaptTrn(\(D_{train}\)) | adaptTrn(\(D_{test}\)) |     |     |      |     |

**Table 2** Comparison of text classification accuracies using all training data
not an ensemble method and “-” reflects this difference in the mixing ratio and ID cells. We used PoE to integrate the three ensemble results (A to C) of the linear sum and each linear sum setting is presented in two rows. The detailed experimental settings for EDA and PoE are discussed in Sections 4.4 and 4.6, respectively. The final ensemble source of PoE was trained on translated data using RoBERTa, as described in Section 4.5.

EMMRT improves the accuracy by 13.6 points for P@1, 13.8 points for P@3, and 4.4 points for P@20, compared to the baseline method. EMMRT also outperforms the fine-tuned BERT without data augmentation by a significant margin \( (p < 0.001) \) based on the Wilcoxon signed-rank test. EDA cannot improve the accuracy of fine-tuned BERTs without data augmentation.\(^{11}\) PoE works well when it is used to integrate the results of a linear sum. PoE improves the P@20 accuracy, but the calculation cost is three times greater than that of EMMRT.

### 4.3 Ablation Studies

In this subsection, we report the results of ablation studies using LSTM-based and BERT-based models. We prepared a general NMT model trained using only JParaCrawl and used the same neural network architecture as the baseline method for classification. For comparison to the baseline method, we trained an LSTM-based model for 50 epochs using all training data. To confirm the effects of EMMRT, we changed the experimental settings one by one from EMMRT to the baseline method. Oversampling is denoted by oversample\((D)\) in the table, which increases the number of instances by the same amount as N\(\times\)M-best round-trip translation performed on the training data. The results of the ablation study for the LSTM-based model are presented in Table 3, whose format is the same as that of Table 2 (“baseline+DS+DA” corresponds to the same model as EMMRT, except for the use of LSTM instead of BERT). In Table 3, DS indicates data separation, meaning an ensemble of two models separately trained using either the first or second set of training data (designated in the ID column in Table 3), whereas the vanilla baseline is an ensemble of two models, both of which were trained on a mixture of two types of training data. DA indicates domain adaptation and OS indicates oversampling.

In our experiment on the LSTM-based model, the oversampled training data improved classification accuracy. The accuracy was also improved by the domain adaptation of NMT and by handling augmented data separately. The results from models that combined the original data with augmented data were worse than those of models using separate data. Compared to the results in Table 2, the most effective improvement was obtained by using BERT, which improved

---

\(^{11}\) We also confirmed the experiment settings for fine-tuned BERT without an ensemble and fine-tuned BERT + EDA with an ensemble, but their accuracy did not exceed the accuracy of EMMRT.
P@1 by more than 8 points.

Similar to the LSTM-based model, the results of an ablation study on the BERT-based model are presented in Table 4, whose format is the same as that of Table 3. It is worth noting that these scores cannot be directly compared to those in Table 3 because stratified 10-fold training data were used in these experiments.

Similar to the LSTM-based model, accuracy was improved by domain adaptation and by handling augmented data separately. In contrast to the experiment on LSTM, adjusting the

| Experiment setting   | method  | 1st:2nd ratio | ID | Training data | Test data | P@1 | P@3 | P@20 | MRR |
|----------------------|---------|---------------|----|---------------|-----------|-----|-----|------|-----|
| MTRRT (baseline)     | LSTM    | 2:1           | 1st | {D_train ∪ RT(D_train)} | D_test | 54.94 | 69.56 | 86.36 | -   |
| baseline             | LSTM    | 7:3           | 1st | {D_train ∪ RT(D_train)} | D_test | 57.71 | 73.32 | 86.76 | 0.671|
| baseline+DS          | LSTM    | 6:4           | 1st | D_train       | D_test | 60.08 | 73.91 | 85.77 | 0.681|
| baseline+DS+DA       | LSTM    | 6:4           | 2nd | adaptRTn(D_train) | adaptRTn(D_test) | 60.28 | 73.52 | 86.36 | 0.683|
| baseline+DS+OS       | LSTM    | 9:1           | 1st | oversample(D_train) | D_test | 61.86 | 74.90 | 87.94 | 0.696|
| baseline+DS+DA+OS    | LSTM    | 7:3           | 2nd | adaptRTn(D_train) | adaptRTn(D_test) | 63.83 | 74.11 | 86.56 | 0.702|

**Table 3** Ablation study on LSTM-based model using all training data

| Experiment setting   | method  | 1st:2nd ratio | ID | Training data | Test data | P@1 | P@3 | P@20 | MRR |
|----------------------|---------|---------------|----|---------------|-----------|-----|-----|------|-----|
| EMMRT (ours)         | jBERT   | 8:2           | 1st | D_train       | D_test | 62.75 | 78.79 | 89.31 | 0.718|
| EMMRT-DA             | jBERT   | 8:2           | 2nd | adaptRTn(D_train) | adaptRTn(D_test) | 62.61 | 78.85 | 89.43 | 0.717|
| EMMRT-DA-DS          | jBERT   | 9:1           | 1st | {D_train ∪ RT(D_train)} | D_test | 61.86 | 77.33 | 89.15 | 0.708|
| EMMRT-DA+OS          | jBERT   | 7:3           | 2nd | adaptRTn(D_train) | adaptRTn(D_test) | 62.63 | 77.27 | 87.33 | 0.710|
| EMMRT + OS           | jBERT   | 7:3           | 1st | oversample(D_train) | D_test | 63.16 | 77.85 | 87.27 | 0.714|

**Table 4** Ablation study on BERT using stratified 10-fold data
number of instances by oversampling improved the P@1 score, but failed to improve the P@3, P@20, and MRR scores.

4.4 Comparison to Other Augmentation Methods

We compared EMMRT to two other data augmentation methods: fine-tuned BERT and EDA (Wei and Zou 2019). For fine-tuned BERT, we fine-tuned two pre-trained BERT models using the original training data and used an ensemble of the two models with a mixing ratio of 0.5. For EDA, we used the original EDA source code. EDA provides four simple operations for data augmentation in English and all of them have positive effects on prediction. We adopted a random swap from EDA because this operation does not depend on WordNet and is applicable to Japanese text, and half of the EDA operations are based on WordNet (Miller 1995) to obtain synonyms. We then ensembled the augmented data using EDA instead of round-trip translated data. In Table 5, EDA(D_train) denotes the randomly swapped stratified 10-fold training data.

Applying the EDA and EDA + ensemble methods yielded similar accuracy to the fine-tuned BERT without data augmentation, but the scores were inferior to those of EMMRT according to the Wilcoxon signed rank test (p < 0.001). This experiment revealed that the ensemble component of EMMRT can also be effective with other data augmentation methods and that EDA can be used as an alternative ensemble method.

4.5 Combination with Other Data

We examined the extent to which accuracy can be improved by EMMRT if fewer data are available. Table 6 presents the classification accuracy of the fine-tuned BERTs when the amount of training data is reduced.

| Experiment setting | method | 1st/2nd ratio | ID Training data | Test data | P@1 | P@3 | P@20 | MRR |
|--------------------|--------|---------------|-----------------|-----------|-----|-----|------|-----|
| EMMRT (ours)      | jBERT  | 8:2           | 1st D_train     | D_test    | 62.75 | 78.79 | 89.31 | 0.718 |
|                    |        |               | 2nd adaptRTn(D_train) | adaptRTn(D_test) |       |       |       |     |
| Fine-tuned BERT    | jBERT  | 5:5           | 1st D_train     | D_test    | 62.25 | 78.34 | 89.01 | 0.714 |
|                    |        |               | 2nd D_train     | D_test    |       |       |       |     |
| EDA                | jBERT  | -             | -               | {D_train ∪ EDA(D_train)} | D_test | 62.75 | 77.77 | 88.70 | 0.714 |
| EDA+ensemble      | jBERT  | 8:2           | 1st D_train     | D_test    | 62.09 | 78.70 | 89.74 | 0.714 |
|                    |        |               | 2nd EDA(D_train) | D_test    |       |       |       |     |

Table 5 Comparison to EDA using stratified 10-fold data

---

12 https://github.com/jasonwei20/eda_nlp
of training data is reduced. $D_{\text{train}1K}$ indicates that the amount of training data is reduced to approximately 1,000 instances and $D_{\text{train}2K}$ indicates that it is reduced to approximately 2,000 instances. In both cases, EMMRT improves all P@k scores and improves the MRR score by more than 0.004 points.

Additionally, we evaluated methods using English language representation models and English augmented data translated by the ja-en translator instead of Japanese language representation models and Japanese data augmented by round-trip translation. Because various pre-trained language representation models are available for English, we used BERT, XLNet, RoBERTa, ALBERT, and ELECTRA. We used the Simple Transformers implementation in the same manner as the other experiments. The ensemble results are listed in Table 7. $\text{adaptTn}(D)$ denotes the stratified 10-fold translated training data, which are increased by N-best translation with domain-adapted NMT. We omitted the results from ALBERT and ELECTRA because they could not improve accuracy.

| Experiment setting | method | 1st:2nd ratio | ID | Training data | Test data | P@1 | P@3 | P@20 | MRR |
|--------------------|--------|---------------|----|--------------|-----------|-----|-----|------|-----|
| less data (1K)     | jBERT  | 8:2           | 1st | $D_{\text{train}1K}$ | $D_{\text{test}}$ | 20.55 | 32.41 | 44.66 | 0.281 |
|                    |        |               | 2nd | adaptRTn($D_{\text{train}1K}$) | adaptRTn($D_{\text{test}}$) |     |     |      |      |
| less data (2K)     | jBERT  | 9:1           | 1st | $D_{\text{train}2K}$ | $D_{\text{test}}$ | 38.34 | 48.42 | 60.67 | 0.447 |
|                    |        |               | 2nd | adaptRTn($D_{\text{train}2K}$) | adaptRTn($D_{\text{test}}$) |     |     |      |      |

Table 6 Combination with less data

| Experiment setting | method | 1st:2nd ratio | ID | Training data | Test data | P@1 | P@3 | P@20 | MRR |
|--------------------|--------|---------------|----|--------------|-----------|-----|-----|------|-----|
| EMMRT (ours)       | jBERT  | 8:2           | 1st | $D_{\text{train}}$ | $D_{\text{test}}$ | 62.75 | 78.79 | 89.31 | 0.718 |
|                    |        |               | 2nd | adaptRTn($D_{\text{train}}$) | adaptRTn($D_{\text{test}}$) |     |     |      |      |
| Fine-tuned BERT    | jBERT  | 5:5           | 1st | $D_{\text{train}}$ | $D_{\text{test}}$ | 62.25 | 78.34 | 89.01 | 0.714 |
|                    |        |               | 2nd | $D_{\text{train}}$ | $D_{\text{test}}$ |     |     |      |      |
| translated data    | jBERT  | 8:2           | 1st | $D_{\text{train}}$ | $D_{\text{test}}$ | 62.49 | 78.36 | 89.15 | 0.715 |
| (BERT)             |        |               | 2nd | adaptRTn($D_{\text{train}}$) | adaptRTn($D_{\text{test}}$) |     |     |      |      |
| translated data    | jBERT  | 8:2           | 1st | $D_{\text{train}}$ | $D_{\text{test}}$ | 62.09 | 78.48 | 89.60 | 0.714 |
| (XLNet)            |        |               | 2nd | adaptRTn($D_{\text{train}}$) | adaptRTn($D_{\text{test}}$) |     |     |      |      |
| translated data    | jBERT  | 8:2           | 1st | $D_{\text{train}}$ | $D_{\text{test}}$ | 62.43 | 78.77 | 89.25 | 0.716 |
| (RoBERTa)          |        |               | 2nd | adaptRTn($D_{\text{train}}$) | adaptRTn($D_{\text{test}}$) |     |     |      |      |

Table 7 Comparisons using 10-fold translated data and English language representation models.
Similar to round-trip translation, the classification accuracy of the ensemble results for BERT, XLNet, and RoBERTa, when fine-tuned with translated data, is higher than that of the fine-tuned BERT without data augmentation or ensembling. Although it is not reflected in the table, domain adaptation improves the classification accuracy of the translated data and has a good effect on using ensembles of prediction results. For example, BERT and XLNet are improved by approximately three points in terms of P@1. In particular, RoBERTa significantly improves the accuracy (approximately 9 points at P@1) with domain adaptation. This indicates that the improvement in NMT translation quality facilitates the use of RoBERTa and increases the availability of language representation models.

4.6 Combination with Other Ensemble Methods

We also investigated whether multiple ensembles could improve accuracy. Based on the large number of potential combinations, we used PoE (Hinton 2002) to aggregate multiple results. In our experiments, the linear sum worked better than PoE for integrating predictions from the models trained on the original data and augmented data, but PoE worked well with linear sum ensemble results because PoE is good at integrating similar predictions. This implies that the characteristics of predictions using round-trip translated data differ from those of predictions using the original data. The linearly summed predictions of both types of data with different ratios are similar. Therefore, we used PoE to integrate the ensemble results produced by the linear sum. The sources used for PoE can be summarized as follows.

A. Use a linear sum ensemble to integrate the results of predictions from the original data and N-best round-trip translated data with domain adaptation using the fine-tuned BERT. The formulation is

\[
\text{Ensemble(Linear}(8:2)\text{, Predict(Model(jBERT, D}_{\text{train}}), D_{\text{test}}),
\]

\[
\text{Predict(Model(jBERT, adaptRT}_{n}(D_{\text{train}})), \text{adaptRT}_{n}(D_{\text{test}}))}.
\]

B. Use a linear sum ensemble to integrate the results of predictions from the original data and N-best round-trip translated data without domain adaptation using the fine-tuned BERT. The formulation is

\[
\text{Ensemble(Linear}(8:2)\text{, Predict(Model(jBERT, D}_{\text{train}}), D_{\text{test}}),
\]

\[
\text{Predict(Model(jBERT, RT}_{n}(D_{\text{train}})), \text{RT}_{n}(D_{\text{test}}))}.
\]

C. Use a linear sum ensemble to integrate the results of predictions from the original data
using the fine-tuned BERT and predictions from the N-best translated data with domain adaptation using the fine-tuned RoBERTa. The formulation is

\[
\text{Ensemble(Linear}(8:2)\), \text{Predict(Model}(j\text{BERT}, D_{\text{train}}), D_{\text{test}}),
\]

\[
\text{Predict(Model}(\text{RoBERTa}, \text{adaptTn}(D_{\text{train}})), \text{adaptTn}(D_{\text{test}})).
\]

We selected these settings to confirm whether a synergistic effect could be obtained from the combination of various models that performed well in the previous experiment. The results of PoE with stratified 10-fold training data are presented in Table 8.

According to these results, PoE performs poorly when it is used instead of a linear sum, but works well when it is used to integrate the results from a linear sum. P@3 and P@20 are improved by combining the ensemble results from multiple data and models. Although it is not reflected in the table, we also attempted to use an ensemble for integrating EDA results (from Table 5) and found that this also improved accuracy slightly. However, the accuracy at low epochs was lower than that before the calculation of PoE, so PoE should be used only after learning has converged. Furthermore, the combination of more predictions makes interpretation more difficult and increases the amount of work required to update a model, so the best balance for practical use should be considered. We judged that round-trip translation and ensemble are the best combinations for ERP packages because interpretation can be performed in Japanese and it is possible to increase the amount of training data while continuing to use a model. In future work,

| Experiment setting          | method  | 1st:2nd ratio | ID Training data | Test data | P@1 | P@3 | P@20 | MRR   |
|-----------------------------|---------|---------------|------------------|-----------|-----|-----|------|-------|
| EMMRT (ours)                | jBERT   | 8:2           | 1st $D_{\text{train}}$ $D_{\text{test}}$ |           | 62.75 | 78.79 | 89.31 | 0.718 |
|                             |         |               | 2nd adaptTn($D_{\text{train}}$) adaptTn($D_{\text{test}}$) |           |      |      |      |       |
| Fine-tuned BERT             | jBERT   | 5:5           | 1st $D_{\text{train}}$ $D_{\text{test}}$ |           | 62.25 | 78.34 | 89.01 | 0.714 |
|                             |         |               | 2nd $D_{\text{train}}$ $D_{\text{test}}$ |           |      |      |      |       |
| PoE instead of Linear       | jBERT   | -             | 1st $D_{\text{train}}$ $D_{\text{test}}$ |           | 58.52 | 75.24 | 86.62 | 0.714 |
|                             |         |               | 2nd adaptTn($D_{\text{train}}$) adaptTn($D_{\text{test}}$) |           |      |      |      |       |
| EMMRT (ours) + PoE for ensemble | A jBERT | 8:2           | 1st $D_{\text{train}}$ $D_{\text{test}}$ |           | 62.67 | 79.11 | 89.47 | 0.718 |
|                             |         |               | 2nd adaptTn($D_{\text{train}}$) adaptTn($D_{\text{test}}$) |           |      |      |      |       |
|                             | B jBERT | 8:2           | 1st $D_{\text{train}}$ $D_{\text{test}}$ |           | 62.67 | 79.11 | 89.47 | 0.718 |
|                             |         |               | 2nd RTn($D_{\text{train}}$) RTn($D_{\text{test}}$) |           |      |      |      |       |
|                             | C jBERT | 8:2           | 1st $D_{\text{train}}$ $D_{\text{test}}$ |           | 62.67 | 79.11 | 89.47 | 0.718 |
|                             |         |               | 2nd adaptTn($D_{\text{train}}$) adaptTn($D_{\text{test}}$) |           |      |      |      |       |

**Table 8** Comparison and combination of PoE and 10-fold data
additional ensemble methods could be used to combine various models. For example, Scikit-learn's StackingClassifier\textsuperscript{13} would be useful for this purpose.

4.7 Discussion

In this subsection, we evaluate the classification accuracy of the proposed method from the perspective of its practical use for ERP package system integration and describe its prospects for additional benefits. We received a response from experienced system designers that “83% of P@3 is useful for work.” This indicates that EMMRT exceeds the target accuracy of P@3=80% and achieves sufficient accuracy for practical use. As mentioned in Section 1, after referring to the suggestions from EMMRT, system designers conduct subsequent work using the suggested components by selecting and operating them to check whether the components satisfy the requirements of customers. Improving the accuracy of P@1 and P@3 compared to the baseline method leads to a reduction in the time required for subsequent work and improves the quality of work. Additionally, we received positive feedback that the proposed method can speed up the training of system designers by helping them understand and notice related functions.

5 Analysis

First, we confirmed the relationship between accuracy improvement and the number of instances in the training data. We divided all classes into three groups: classes containing five or fewer instances, classes containing 6 to 10 instances, and classes containing 11 or more instances. Compared to fine-tuned BERT without an ensemble and round-trip translation, EMMRT improved the MRR of classes containing five or fewer instances in the training data without reducing the MRR of classes containing 11 or more instances in the training data.

Second, we analyzed the predictions based on round-trip translated data to check the number of classes whose classification accuracies changed. Because linear summation produced a better result than PoE in ensembles of predictions based on round-trip translated data, the characteristics of the round-trip translation data appear to be different from those of the original data. According to the differences between the predictions from the original data and those from the round-trip translated data, we analyzed how round-trip translation affects accuracy. We also analyzed the classes and instances that were significantly different before and after the domain adaptation of NMT. Because the data considered in this study contained a large number of

\textsuperscript{13} https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html
classes, over half of the classes contained five or fewer instances, so we counted the number of times the correct labels (P@1) were predicted during 10-fold iterations. The number of correct labels varied between 0 and 10 because in the worst case, none of the 10-fold models would predict the correct label and in the best case, all of the 10-fold models would predict the correct label. We then calculated the differences between the models. The results are listed in Table 9.

Table 9 presents the model performance before adaptation, model performance after adaptation, and differences between the two results from left to right. The comparison source and comparison destination are described by the training data and testing data that they use. The comparison results are divided into stages from “>+5” to “<-5,” where “>+5” indicates that the comparison destination is dramatically improved and “<-5” indicates that the comparison destination is dramatically deteriorated compared to the source. For example, the case in which the correct label is predicted in 2 of 10 iterations in the source and 8 of 10 iterations in the destination is categorized as “>+5.” The results of this analysis demonstrate that classification based on round-trip translated data is inferior to that based on original data, but new correct answers are acquired in some cases. A comparison between the results before and after domain adaptation reveals both improvement and deterioration, but the improvement is greater than the deterioration. This confirms that the superior prediction results from round-trip translated data also apply to ensemble results.

We also compared the unique word n-grams in the original data and round-trip translated data to examine whether the sparsity and diversity problems were alleviated. For diversity, we examined how many new n-grams that were not in the original data were generated by round-trip translation. For sparsity, we calculated the average number of shared n-grams between the training data and round-trip translated data for each class of ERP components. The results are listed in Table 10.

As a result of round-trip translation, the number of unique n-grams in the training data decreased and those of the testing data increased. In the training data, approximately 31% and 47% of 1-grams and 2-grams were new expressions, respectively, and approximately 45% and 70%

| before adaptation | after adaptation | the difference of the two results |
|-------------------|------------------|----------------------------------|
| Training data     | Test data        | Training data                    | Test data |
| $D_{train}$       | $D_{test}$       | adaptRTn($D_{train}$)            | adaptRTn($D_{test}$) |
|                   |                  | 4                                | 15         |
|                   |                  | -3 to -5                         | 51         |
|                   |                  | 4 to +3                          | 83         |
| $RTn(D_{train})$  | $RTn(D_{test})$  | adaptRTn($D_{train}$)            | adaptRTn($D_{test}$) |
|                   |                  | 64                               | 53         |
|                   |                  | -3 to -5                         | 36         |
|                   |                  | 4 to +3                          | 30         |

Table 9 Differences in number of correct predictions (P@1) between two models using 10-fold iterations
of 1-grams and 2-grams in the testing data were new expressions, respectively. The number of shared n-grams between training data and testing data per class of ERP components increased by approximately 3% for 1-grams and decreased by approximately 22% for 2-grams. In particular, in classes where the accuracy was improved during the 10-fold iterations, the number of shared n-grams per class increased significantly \( p < 0.001 \) according to the Mann-Whitney U test.\(^{14}\) In contrast, in classes where the number of shared n-grams per class increased, the accuracy during the 10-fold iterations did not improve in all cases. These results indicate that round-trip translation produces new and more similar expressions for training data and testing data. Therefore, round-trip translation alleviates the problems of diversity and sparsity in some cases. For example, the original expression “資金繰り表” (shikin guri hyo) ‘a cash flow table’ is changed to “資金スケジュール” (shikin sukejuru) ‘a schedule of funding’ by round-trip translation and another original expression “入出金予定” (nyusyukkin yotei) ‘a plan of receiving and making payment’ is changed to “入出金スケジュール” (nyusyukkin sukejuru) ‘a schedule of receiving and making payment.’ Both of the original expressions are related to cash flow and funding management, and new expressions are relatively similar. Conversely, in other cases, expressions were changed to new expressions that had different meanings by round-trip translation and the classification accuracy deteriorated. For example, “照会” (sho-kai) ‘refer’ was changed to “問い合わせ” (toiawase) ‘inquiry’ because “照会” is also used to express inquiry. Consequently, in some cases, more accurate predictions that could not be derived from the original data were produced from round-trip translated data and the ensemble model took advantage of both types of results.

\(^{14}\) The same trends were observed in shared 3-grams and 4-grams.
6 Conclusion

To support system designers in ERP package integration, we improved the accuracy of the automatic suggestion of software components. To improve accuracy, we applied data augmentation through round-trip translation to both training data and testing data using a domain-adapted NMT and proposed an effective ensemble method for round-trip translated data. As a result, we achieved a sufficient level of accuracy for practical use.

The proposed method can also be applied to text classification tasks other than ERP component suggestion and may improve accuracy in various business areas where accuracy is currently inadequate based on a lack of data for machine learning.

This study demonstrated that machine translation, which has already been used to support human-to-human communication, can also be used to support human-machine interaction. We found that NLP can even benefit areas with limited data by using a large-scale trained model with rich text data, such as BERT, and using NMT round-trip translation for data augmentation.

Acknowledgement

We are very grateful to Keiichi Zukeran, Masaaki Abe, and Ryuzo Horiguchi, who are SAP consultant experts at FUJITSU LIMITED and provided intensive support for the arrangement of experiments, including gathering data, labeling training data, and advising on SAP-related resources. They shared a variety of business knowledge related to ERP and contributed significantly to the development and evaluation of our system.

References

Almarimi, N., Ouni, A., Bouktif, S., Mkaouer, M. W., Kula, R. G., and Saied, M. A. (2019). “Web Service API Recommendation for Automated Mashup Creation using Multi-objective Evolutionary Search.” *Applied Soft Computing*, 85. 105830.

Berard, A., Calapodescu, I., and Roux, C. (2019). “Naver Labs Europe’s Systems for the WMT19 Machine Translation Robustness Task.” In *Proceedings of the 4th Conference on Machine Translation*, Vol. 2, pp. 526–532. Association for Computational Linguistics.

Breiman, L. (1996). “Stacked Regressions.” *Machine Learning*, 24 (1), pp. 49–64.

Chen, C. and Xing, Z. (2016). “SimilarTech: Automatically Recommend Analogical Libraries across Different Programming Languages.” In *Proceedings of the 31st IEEE/ACM Inter-
national Conference on Automated Software Engineering (ASE), pp. 834–839. Association for Computing Machinery.

Chu, C. and Wang, R. (2018). “A Survey of Domain Adaptation for Neural Machine Translation.” In Proceedings of the 27th International Conference on Computational Linguistics, pp. 1304–1319. Association for Computational Linguistics.

Clark, K., Luong, M., Le, Q. V., and Manning, C. D. (2020). “ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators.” In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020.

Coscrato, V., Inacio, M. H. d. A., and Izbicki, R. (2020). “The NN-Stacking: Feature Weighted Linear Stacking Through Neural Networks.” Neurocomputing, 399, pp. 141–152.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” In Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 4171–4186.

Hinton, G. E. (2002). “Training Products of Experts by Minimizing Contrastive Divergence.” Neural Comput, 14 (8), pp. 1771–1800.

Hu, J. E., Rudinger, R., Post, M., and Van Durme, B. (2019a). “PARABANK: Monolingual Bitext Generation and Sentential Paraphrasing via Lexically-Constrained Neural Machine Translation.” In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, pp. 6521–6528.

Hu, J. E., Singh, A., Holzenberger, N., Post, M., and Van Durme, B. (2019b). “Large-Scale, Diverse, Paraphrastic Bitexts via Sampling and Clustering.” In Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL), pp. 44–54. Association for Computational Linguistics.

Kobayashi, S. (2018). “Contextual Augmentation: Data Augmentation by Words with Paradigmatic Relations.” In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pp. 452–457. Association for Computational Linguistics.

Koehn, P. and Knowles, R. (2017). “Six Challenges for Neural Machine Translation.” In Proceedings of the 1st Workshop on Neural Machine Translation, pp. 28–39. Association for Computational Linguistics.

Kohavi, R. (1995). “A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection.” In Proceedings of the 14th International Joint Conference on Artificial
Intelligence - Volume 2, pp. 1137–1143.

Kudo, T. and Richardson, J. (2018). “SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing.” In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 66–71.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2020). “ALBERT: A Lite BERT for Self-supervised Learning of Language Representations.” In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020.

Liu, P., Wang, X., Xiang, C., and Meng, W. (2020). “A Survey of Text Data Augmentation.” In 2020 International Conference on Computer Communication and Network Security (CCNS), pp. 191–195.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). “RoBERTa: A Robustly Optimized BERT Pretraining Approach.” CoRR, abs/1907.11692.

Lu, X., Zheng, B., Velivelli, A., and Zhai, C. (2006). “Enhancing Text Categorization with Semantic-enriched Representation and Training Data Augmentation.” Journal of the American Medical Informatics Association, 13 (5), pp. 526–535.

Mabert, V., Soni, A., and Venkataramanan, M. (2000). “Enterprise Resource Planning Survey of U.S. Manufacturing Firms.” Production and Inventory Management Journal, 41, pp. 52–58.

Madakam, S., Holmukhe, R. M., and Jaiswal, D. K. (2020). “The Future Digital Work Force: Robotic Process Automation (Rpa).” JISEM - Journal of Information Systems and Technology Management (Online), 16 (0). e201916001.

Maniwa, S., Awatsu, M., Okada, I., Ideuchi, M., Oida, Y., and Wakasugi, K. (2017). “Application of SE Reforms Utilizing AI Technology.” Fujitsu, 68 (6), pp. 75–83.

Matsuda, H. (2020). “GiNZA - Practical Japanese analysis based on Universal Dependencies [Translated from Japanese].” Journal of Natural Language Processing, 27, pp. 695–702.

Miller, G. A. (1995). “Wordnet : A Lexical Database for English.” Communications of the ACM, 38 (11), pp. 39–41.

Morishita, M., Suzuki, J., and Nagata, M. (2020). “JParaCrawl: A Large Scale Web-Based English-Japanese Parallel Corpus.” In Proceedings of the 12th Language Resources and Evaluation Conference, pp. 3603–3609.

Nakamura, S., Hieka, K., Okada, I., Saito, M., Oida, Y., Masuda, K., Abe, M., and Horiguchi, R. (2014). “Development of Design Know-how Cases Retrieval Support System on a Package
System which Corresponds to Customer Requirements [Translated from Japanese].” The Japanese Society for Artificial Intelligence Special Interest Group on Knowledge and Skills Transfer (SIG-KST).

National Institute of Information and Communications Technology (2020). “NICT BERT Japanese Pre-trained model [Translated from Japanese].” https://alaginrc.nict.go.jp/nict-bert/.

Ohga, T., Yano, M., and Ohnishi, M. (2018). “Verification of Ensemble Learning by Data and Test Time Augmentation Effects (Pattern Recognition / Media Comprehension) [Translated from Japanese].” IEICE Technical Report, 117 (391), pp. 135–140.

Olhager, J. and Selldin, E. (2003). “Enterprise Resource Planning Survey of Swedish Manufacturing Firms.” European Journal of Operational Research, 146 (2), pp. 365–373.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. (2019). “fairseq: A Fast, Extensible Toolkit for Sequence Modeling.” In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), pp. 48–53.

Ott, M., Edunov, S., Grangier, D., and Auli, M. (2018). “Scaling Neural Machine Translation.” In Proceedings of the 3rd Conference on Machine Translation: Research Papers, pp. 1–9. Association for Computational Linguistics.

Ouni, A., Kula, R. G., Kessentini, M., Ishio, T., German, D. M., and Inoue, K. (2017). “Search-based Software Library Recommendation using Multi-objective Optimization.” Information Software Technology, 83, pp. 55–75.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). “Bleu: a Method for Automatic Evaluation of Machine Translation.” In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ragni, A., Knill, K., Rath, S., and Gales, M. (2014). “Data Augmentation for Low Resource Languages.” In Proceedings of INTERSPEECH, pp. 810–814.

Sagi, O. and Rokach, L. (2018). “Ensemble Learning: A Survey.” Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8, e1249.

Sakamoto, Y., Oida, Y., Ideuchi, M., Okada, I., Utiyama, M., Higashiyama, S., Ono, J., and Sumita, E. (2018). “Accuracy Improvement of ERP Package Parts Recommendation using Data Augmentation Method by Neural Machine Translation [Translated from Japanese].” The Japanese Society for Artificial Intelligence Special Interest Group on Knowledge and Skills Transfer (SIG-KST).
Shibata, T., Kawahara, D., and Kurohashi, S. (2019). “Accuracy Improvements in Japanese Syntactic Analysis using BERT [Translated from Japanese].” In Proceedings of the 25th Annual Meeting of the Association for Natural Language Processing, pp. 205–208.

Song, H., Dabre, R., Fujita, A., and Kurohashi, S. (2020). “Coursera Corpus Mining and Multistage Fine-Tuning for Improving Lectures Translation.” In 12th Language Resources and Evaluation Conference, pp. 3640–3649. European Language Resources Association.

Suzuki, M. (2019). “Pretrained Japanese BERT models.” https://github.com/cl-tohoku/bert-japanese.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). “Attention Is All You Need.” Advances in Neural Information Processing Systems, pp. 5998–6008.

Wei, J. and Zou, K. (2019). “EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks.” In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 6382–6388.

Wieting, J. and Gimpel, K. (2018). “ParaNMT-50M: Pushing the Limits of Paraphrastic Sentence Embeddings with Millions of Machine Translations.” In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 451–462. Association for Computational Linguistics.

Wieting, J., Mallinson, J., and Gimpel, K. (2017). “Learning Paraphrastic Sentence Embeddings from Back-Translated Bitext.” In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 274–285. Association for Computational Linguistics.

Wolpert, D. H. (1992). “Stacked Generalization.” Neural Networks, 5, pp. 241–259.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. (2019). “XLNet: Generalized Autoregressive Pretraining for Language Understanding.” In Advances in Neural Information Processing Systems 32 (NeurIPS 2019), pp. 5753–5763. Curran Associates, Inc.

Yu, A. W., Dohan, D., Luong, M.-T., Zhao, R., Chen, K., Norouzi, M., and Le, Q. V. (2018). “QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension.” CoRR, abs/1804.09541.

Zhou, D. X. and Resnick, P. (2009). “Assessment of Conversation Co-mentions as a Resource for Software Module Recommendation.” In Proceedings of the 3rd ACM Conference on Recommender Systems, pp. 133–140. Association for Computing Machinery.
Masao Ideuchi: He received a B.S. degree in Environment and Information Studies from Keio University, Japan, in 2004, and an M.S. degree from the Graduate School of Media and Governance of the Cyber Informatics Program of Keio University, Japan, in 2006. From 2006 to 2020, he worked on the development of products and services related to natural language processing at FUJITSU LIMITED and the Fujitsu Research Institute. He has been a member of the National Institute of Information and Communications Technology (NICT) and a Ph.D. student at the Nara Institute of Science and Technology (NAIST) since 2020.

Yohei Sakamoto: He received an M.S. degree in Particle Physics at Kyoto University, Japan, in 2013. Currently, he is a data scientist at Ridgelinez Ltd., proposing and supporting PoC for AI implementation projects and business process improvement in companies.

Yoshiaki Oida: He received a B.S. degree in Engineering and a M.S. degree in Human and Engineered Environmental Studies from the University of Tokyo in 2010 and 2012, respectively. He worked on the development of a discrete event simulator for resource management across multiple information system development projects and earned a Ph.D. from the Graduate School of Frontier Sciences, the University of Tokyo, in 2019. He is currently a managing engineer at FUJITSU LIMITED and since 2020, he has been a part-time lecturer in the Faculty of Engineering of the University of Tokyo. His research interests include model-based systems engineering in the manufacturing and information systems industries.

Isaac Okada: He received his Ph.D. in Science from the Graduate School of Frontier Sciences, the University of Tokyo, in 2019. He is now the Executive Engineer of the Digital Transformation Business Unit at FUJITSU LIMITED, a Visiting Professor in the Faculty of Commerce of Senshu University, Special Lecturer in the Faculty of Engineering of the University of Tokyo, and Part-time Lecturer at the School of Computing of the Tokyo Institute of Technology. His research interests include system dynamics, artificial intelligence (machine learning), natural language processing, knowledge management technologies, semantic technologies, and metadata for business.

Shohei Higashiyama: He received a B.S. degree in Engineering and M.S. degree in System Informatics from Kobe University in 2012 and 2014, respectively. He
was a researcher at the NEC Corporation from 2014 to 2019. He is currently a technical researcher at the National Institute of Information and Communications Technology (NICT) and Ph.D. student at the Nara Institute of Science and Technology (NAIST).

**Masao Utiyama**: He is an executive researcher at the National Institute of Information and Communications Technology, Japan. He completed his doctoral dissertation at the University of Tsukuba in 1997. His main research field is machine translation.

**Eiichiro Sumita**: He is an NICT fellow, the Associate Director-General of the Advanced Speech Translation Research and Development Promotion Center (ASTREC), and also serves as the president of the International Association for Machine Translation (IAMT) and the president of the Asia-Pacific Association for Machine Translation (AAMT). His research interests include machine translation and e-learning.

**Taro Watanabe**: He received his B.E. and M.E. degrees in Information Science from Kyoto University, Kyoto, Japan in 1994 and 1997, respectively, and obtained an M.S. degree in language and information technologies from the School of Computer Science, Carnegie Mellon University in 2000. In 2004, he received a Ph.D. in informatics from Kyoto University, Kyoto, Japan. He also worked as a researcher at ATR, NTT, NICT, and Google Inc. He is a professor at the Nara Institute of Science and Technology (NAIST). His research interests include natural language processing, machine learning, and statistical machine translation.

(Received February 26, 2021)
(Revised June 1, 2021)
(Accepted July 3, 2021)