Utilizing Natural Language Processing to Develop a Chat-Based Learning Platform to Support Second Language Acquisition

DISSERTATION

Andre Rusli

Tokyo Denki University

September 2023

Abstract

This study focuses on the exploration and implementation of various natural language processing (NLP) techniques to build a platform that supports language learners in learning and practicing text-based conversations in English as a foreign language (EFL). Along with the advancements of web and mobile technologies, computer-assisted language learning has been gaining traction in recent years. People around the world use a range of technologies connected to the internet to learn and communicate with each other using foreign languages, in complement to formal education at schools and universities. These online interactions are especially required more than usual when the world was hit by the COVID-19 pandemic, which started towards the end of 2019. Since then, the global industry, including schools and universities, has been forced, more than before, to communicate remotely using various internet technologies. Along with the rise of social media and communication platforms, chatting has become a common way of communicating between people, including at workplaces. Tools like Slack, Microsoft Teams, and Discord are increasingly being used as virtual offices in which many work interactions are happening in real-time.

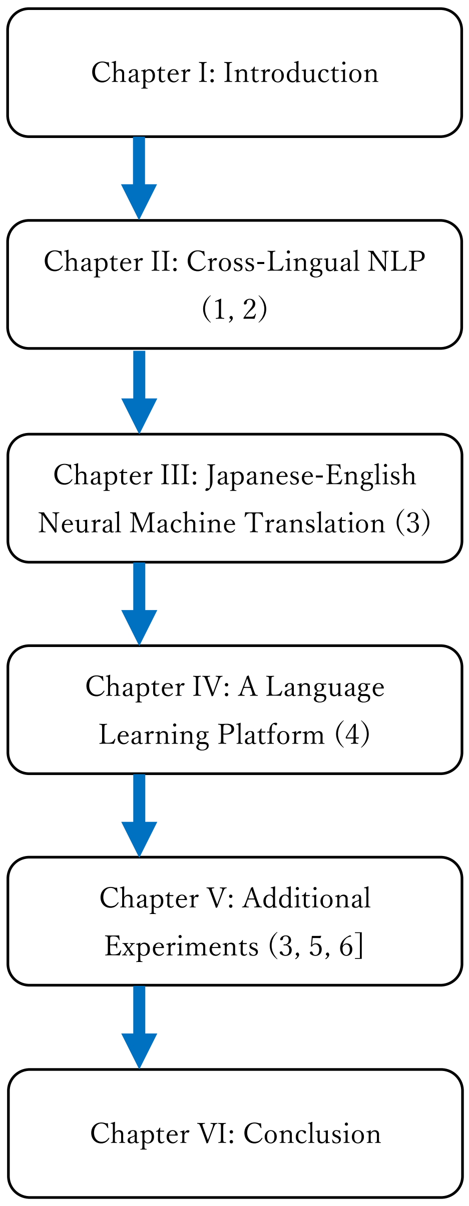
This study aims to address the gap in EFL education, focused on Japanese L1 students, to provide a platform for both teachers to easily integrate text-based conversation practices in classes and students to learn and practice English text conversations. The study focuses on the applicability of recent advances in the natural language processing field, such as morphological analysis tools, Transformers-based models, and various evaluation metrics, to build the system. The proposed system enables the user to conduct self-practice with automatically generated questions and get immediate feedback to conduct self-assessment, especially when they are outside classes and teacher's support is limited. Based on the experiments, the study shows several useful recent advances in the NLP field for supporting second language acquisition, and the proposed system is easy to use and considered useful for intermediate students to practice English. The use of computer-assisted language learning (CALL) and large language models (LLMs) has the potential to revolutionize the way English as a foreign language is learned and taught. Researchers and developers should always put the human first when developing these systems, as no matter how advanced the technologies behind the system are, if they are not used accordingly by the users due to misunderstanding or inability to do so, they will be less meaningful.

Acknowledgment

I would like to express my deepest gratitude to my supervisor, Prof. Makoto Shishido, for his unwavering support, guidance, and encouragement throughout my doctoral journey. Your mentorship and patience have been invaluable to me, and I have learned a great deal from you and hope to continue to apply your wisdom and guidance to my future endeavors. I would also like to thank my fellow research students and colleagues from the laboratory and part-time workplaces who have shared my academic journey. Your camaraderie and support have made this journey more enjoyable and memorable. I am grateful for the countless discussions and ideas shared, and I look forward to future collaborations. I am also grateful to the staff at the university international office for their assistance in making my student life in Japan a truly great experience. Their support and dedication to helping international students are truly commendable.

Finally, I would like to extend my heartfelt appreciation to my family and friends. To my wife, mom, dad, and brother, thank you for your unconditional love, unwavering support, and encouragement throughout achieving my doctoral degree. Your presence in my life has been my greatest source of strength and inspiration. To all my friends in and out of Tokyo, thank you for being the best friends anyone could ask for. Your constant encouragement, laughter, and companionship have been instrumental in making this journey worthwhile.

List of Chapters and Publications


Chapter I: Introduction
Chapter II: Cross-Lingual NLP (1, 2)
Chapter III: Japanese-English Neural Machine Translation (3)
Chapter IV: A Language Learning Platform (4)
Chapter V: Additional Experiments (3, 5, 6]
Chapter VI: Conclusion

(1) Rusli, A. and Shishido, M., 2021. An Experimental Evaluation of Japanese Tokenizers for Sentiment-Based Text Classification. Online, Association of Natural Language Processing.

(2) Rusli, A. and Shishido, M., 2022. On the Applicability of Zero-Shot Cross-Lingual Transfer Learning for Sentiment Classification in Distant Language Pairs. Online, Association of Natural Language Processing.

(3) Rusli, A. and Shishido, M., 2022. Zero-Pronoun Annotation Support Tool for the Evaluation of Machine Translation on Conversational Texts. Journal of Natural Language Processing, 29(2), pp. 493-514.

(4) Rusli, A. and Shishido, M., 2022. An Interactive Learning Platform with Machine Translation for Practicing Text-Based Conversational English. Ise-shi, Mie, Japan, IEEE.

(5) Rusli, A. and Shishido, M., 2023. An Analysis on Automated Metrics for Evaluating Japanese-English Chat Translation. Okinawa, Japan, Association of Natural Language Processing.

(6) *(to be published)* Rusli, A. and Shishido, M., 2023. Development of a Language Learning Platform with Question-Answer Generation and Scoring for Practicing Text-Based Conversations. Vienna, Austria, IAFOR.

Table of Contents

Chapter I Introduction

1.1 Background

In Japan, as with many other countries whose primary language is not English, promoting the importance of English language proficiency often becomes in line with internationalization [1]. Ideally, new graduates from universities across the country should achieve proficiency to be ready to compete in the job market. To accomplish that, nowadays, English has become one of the most important subjects to learn by university students. However, there is still less emphasis on the importance of text-based conversations (e.g., chats), which have increasingly become an essential skill in the industry in recent years. Furthermore, since the Covid-19 pandemic hit the world at the beginning of 2020, many organizations have been severely disrupted, and online education and work have rapidly become popular. Because of this, more interactions between peers, colleagues, teachers, and students are happening online, both in video calls and chats. Some of the most popular communication platforms used in the industry are Slack[1] and Discord[2]. This increase in online interactions means that new graduates would need to adapt to this different style of English communication compared to the more formal ones that were commonly taught in universities.

Most students graduating from higher education institutions, such as universities, have already learned English through courses and other language learning activities. However, to reach high proficiency levels and be successful at interacting with others, there are still many obstacles along this journey. For example, when learners can understand many of the vocabularies in a foreign language, they can infer a lot of words through the context. Many students stop actively using a foreign language and only focus on passively consuming foreign media instead. As a result, there is not enough repetition of the advanced vocabulary that the student needs to learn for it to become part of the productive vocabulary [2]. One possible way to enable the user to practice conversations, especially written ones, is by incorporating translation as a language learning method.

With the recent advances in neural machine translation (NMT) models and computer-assisted technologies, there are more ways to provide the support needed by language learners, each according to the requirements. One of the most notable studies that advance NMT is the Transformer model [3]. Several other studies have proposed the use of context- aware NMT approaches that integrate source- and target-side contexts [4] [5] [6], which could be helpful in chat translation, since the context of a sentence could result in a different translation. In order to support the students to attain proficiency in foreign languages, many researchers and practitioners explore the applicability and usefulness of

[1] https://slack.com

[2] https://discord.com

computer-assisted language learning (CALL) platforms, such as mobile and game applications which are shown to be effective in promoting active study in learning English and encouraging the students to produce language in their own words [7]. Commercial products, such as Duolingo[3] and Clozemaster[4], have also become very popular among foreign language learners.

An essential building block in of recent computer-assisted language learning systems is the use of natural language processing and advanced machine learning algorithms, such as to using pragmatic reasoning to train neural networks for a listener model [8] and automated assessments [9].Amongst these new advancements in the natural language processing field, the arrival of large language models has shown the potential to advance language education even further. One key aspect of these models is the use of the Transformer architectures [10] and self-attention mechanisms [3], which have greatly improved their ability to handle long-range dependencies in natural language texts. The transformer architecture, with its self-attention mechanism, allows the model to better understand the relationships between words in a sentence, regardless of their position.

This has numerous potential applications in education, such as generating practice problems and quizzes to aid students to better understand, contextualize and retain the material they are learning [11]. These models can help both students and teachers by providing access to vast amounts of language data and sophisticated algorithms that can generate practice problems, quizzes, and personalized feedback. For example, teachers can use large language models to automatically generate questions [12], or for automated assessment and scoring [13] [14]. This can save teachers a significant amount of time and provide students with quicker and more personalized feedback. On the other hand, for the students, large language model like ChatGPT can generate interactive dialogs which are personalized to each student, improving the overall experience of foreign language learning [15]. However, despite the potential benefits of these recent technologies, there are still many limitations and inconsistencies [16] of the systems, and also concerns about whether the users are ready to adopt these new technologies, especially to support foreign language learning. Commercial products that use various natural language processing techniques, such as have also become very popular among foreign language learners.

1.2 Research Objectives

Within the context of English learning for L1 Japanese speakers, this research aims to investigate the challenges faced by learners, mainly for intermediate learners to practice text-based conversations. In order to build a useful platform to support English learners practice text-based conversation, this research aims to implement, experiment, and evaluate NLP techniques that can be useful to achieve our intended platform. Finally, using the findings we get from the various

[3] https://duolingo.com/learn

[4] https://clozemaster.com

experimentations, we then propose an interactive web-based software platform, with machine translation, to be used by learners for practicing text conversations in the form of chats and exercises for self-study. The resulting platform enables students to chat with their peers and teachers with Japanese to English translation support and conduct self-study via exercises. The system implements a framework to automatically generate questions given a dataset of parallel conversations and it allows the teachers to manually check and provide individualized feedback to the students. Moreover, we also implemented an automated scoring system which leverages large language models to provide immediate feedback to the students, enabling them to quickly assess the submitted answers.

1.3 Structure of Dissertation

Chapter I serves as an introduction of the motivation, background, and the direction to where our research is heading to. It describes the existing state of research in the computer-assisted language learning (CALL) field. We show some works of research that emphasize about how mobile and web technology could play a huge role in improving students' learning experience, and also how we set the direction to propose and develop the web-based learning platform, which utilizes a variety of natural language processing techniques that focuses on the use of large language models. The proposed platform would provide a chat platform with translation support for users to communicate with each other using either Japanese or English, automatically generate practice questions to reduce the work required by teachers, and also automatically score the student's answers to provide immediate feedback for self-practice. Based on this direction, a series of experimentations is conducted to implement, analyze, and evaluate existing machine learning (ML) and natural language processing (NLP) techniques and methodologies to find out the ones that suit our requirements, which then are used to build the platform.

Our research starts off by looking for the right method to perform morphological analysis of Japanese text documents for our case (**Chapter II**). Unlike alphabetical languages, texts written in Japanese do not contain whitespaces to separate each word or phrase in a sentence. Furthermore, depending on the context, the same combination of characters can be separated (or tokenized) in multiple different ways. We review some existing methods for tokenizing Japanese texts and evaluate their performance on some ML tasks. In addition to this, instead of building one model for each language, recent advances in the NLP field show that it is possible to build a multilingual model. Such language models, if performing as expected, could reduce the time and effort in building a system than is required to handle more than one language, such as the ones often seen in the foreign language learning field. Following the previous experiment, we explore the applicability of such models for to process three languages, English, Japanese, and Indonesian, focusing on the XLM-RoBERTa model which is trained using texts from 100 languages (**Chapter II**).

Using the knowledge we get from the previous experiments, the next one focuses on Japanese-English neural machine translation (NMT) which, at later stages, will be incorporated to the learning platform we are building. In this experiment, we start by surveying existing works of research to get hold of the state of Japanese-English machine translation. Afterwards, we train our own NMT model based on the Transformers architecture and fine-tuned the model for chat translation, which is the focus of the platform we are building for EFL learners and compare its performance with existing open-sourced models that is able to translate sentences from Japanese to English (**Chapter III**). In addition, to improving the field by focusing on the anaphoric zero-pronoun phenomenon which is prevalent in Japanese sentences, especially in a conversational environment. Our work proposes and develops a tool called the Zero-Pronoun Annotation Support Tool (0Past) to make it easier for researchers and practitioners to build a specialized evaluation set to measure the performance of a Japanese-English NMT model in solving the anaphoric zero-pronoun. Using the tool, we annotate a set of Japanese-English parallel text dialogue corpus and use it to re-evaluate several NMT models. Furthermore, we also show that the annotated data can also be helpful to train a zero-pronoun sentence classifier that could detect whether a sentence written in Japanese contains the anaphoric zero-pronoun phenomenon or not.

After the previous experiments related to potential NLP techniques and tools which can be used to build a computer-assisted language learning, we conducted preliminary surveys and interviews with Japanese university students regarding their view on text-based conversation, which is massively used, especially during remote work and study. Based on the insights we get from these surveys; we start designing and developing the language learning platform to enable Japanese student to better practice and improve their skill and confidence in communication using text in English (**Chapter IV**). We build the platform as a web-based tool so that it can be used by many users, both on laptop/PC and mobile devices. The frontend web application is built using VueJS, and the backend system is supported by Google Firebase for authentication and database, and Heroku in which we deploy our APIs that are written in Python, which mainly support processes related to text processing and ML model inference. Towards the end of our series of experiments, we conducted a usability test with actual users in a Japanese university, gather feedback, perform several feature improvements, and finally report the findings. In **Chapter V**, we describe some additional experiments that are conducted in addition to the main experiments previously mentioned.

Chapter II Cross-Lingual Natural Language Processing

This section is mainly divided into two parts in which, firstly, we focus on Japanese tokenizers, and, secondly, the applicability of XLM-RoBERTa in processing cross-lingual texts with zero-shot transfer learning.

2.1 Tokenization for Japanese Texts

In many languages, words are often considered as the basic unit of texts. Several works of research also show that when using n-gram language models, word n-grams are relatively better than character n-grams to convert texts into tokens when building a text classification model [17] [18]. Japanese texts pose different challenges for a machine learning algorithm to perform well. Sentences in Japanese contain no whitespace between words, so the common preprocessing phase to explicitly split words based on whitespaces could not be easily conducted. In addition, the combination of characters in a sentence could vary and may be ambiguous as they could have different meanings depending on the combinations. Many works of research have developed morphological analysis tools for Japanese language. Some of the popular tools for morphological analysis tools for Japanese are including MeCab [19], Sudachi [20], and SentencePiece [21].

In this section, we focus on utilizing one of the many features provided by the previously mentioned tools which is the tokenizer for segmenting Japanese texts by words or sub-words. Furthermore, this research then implements the tokenizer as a preprocessing step towards building supervised sentiment-based text classification models with Term Frequency–Inverse Document Frequency (TF-IDF) vectorization. Regarding text classification tasks for Japanese texts, many recent works experiment with complex models using various deep learning approaches such as BERT [22] [23], Bidirectional LSTM-RNN [24], and Quasi-RNN and Transformer model [25], most are competing to achieve state-of-the-art performance. In our early work presented in this article, we aim to experiment with and report text classification results on baseline traditional algorithms with less computational cost and more interpretability. Two classifiers used in in this experiment are the Multinomial Naïve Bayes and Logistic Regression. Logistic Regression, which used to be the default choice for text classification, can be found used as a baseline linear model for Japanese text classification such as in [22] [26] [27]. Many researchers are also still exploring the potential and possibility of enhancing Multinomial Naïve Bayes' ability as a classification method, such as in [28]. By using these two methods, we emphasize more on the tokenization tools, rather than the classification algorithms. In addition, to also provide reports on the performance of traditional machine learning algorithms which with its limitations could still perform well. The objectives of this experiment are summarized as follows:

a. Implement Japanese text tokenization using MeCab, Sudachi, and SentencePiece, then use the results to build models for binary sentiment-based text classification using TF-IDF with Multinomial Naïve Bayes and Logistic Regression, and
b. Compare and report the performance results in terms of time and error percentages.

2.1.1 Related Works

MeCab, Sudachi, and SentencePiece as morphological analysis tools have been used in many works of research in natural language processing. Although it is not exactly a newly proposed tool (the last update on its GitHub repository[5] is version 0.996 in February 2013), MeCab is still used extensively as a word segmentation tool for preprocessing Japanese text in recent years. One example is the work by Zhang and LeCun [26], in which MeCab is used to segment texts in Japanese and Korean (with additional model in Korean language). Sudachi is another tool that mainly emphasizes the focus on continuous maintenance and feature richness, as it aims to support business use. It is also currently used by spaCy[6], a well-known library in NLP, as a tool for their pretrained statistical model for Japanese.

SentencePiece, on the other hand, is a tool for subword segmentation that uses a different approach. The authors describe SentencePiece as a language-independent subword tokenizer and detokenizer designed for Neural- based text processing [21]. While using a different approach with focus on providing a method to support language-independent multilingual text processing, its performance is shown to be effective for various tasks, such as English-Japanese neural machine translation [21], Japanese news text classification [23], and sentiment analysis in Japanese [22]. However, even though many articles have shown the ability of each tools to be utilized for various tasks in natural language processing, works that provide a hand-in-hand comparison between the features provided by the tools in the same environment and configurations are still hard to find.

2.1.2 Dataset

Our study is based on the Japanese Rakuten product review binary sentiment dataset provided in [26]. The datasets (both train and test set), which are available as CSV files, consist of three columns supposedly the binary sentiment label, review title, and review text, with examples such as the followings:

- Label: 1 (negative)
  Review Title: 臭い
  Review Text: 余りにも、匂いがきつく安物み たいです。¥¥n 安いから仕方ないかな?

[5] http://taku910.github.io/mecab/

[6] https://spacy.io

- Label: 2 (positive)
  Review Title: 早いし安い
  Review Text: 毎回利用しています。納品が早 いし何よりお安く大変便利です。また利用します。

We use the binary sentiment label and review text and we randomly sampled 10% of the provided training and testing data and only used the sampled 10% of the total data in our experiments, that is 340,000 reviews for training and 40,000 reviews for testing as we aim to experiment with many configurations in a limited setting.

2.1.3 System Specification

The experiments are conducted by using Python 3.7 and Jupyter Notebook, running on an Ubuntu virtual machine provided by Google Compute Engine, using a machine type c2-standard-4 with 4 vCPUs and 16 GB memory. However, it is also possible to run all of the experiments on Google Colaboratory. For the tokenization tools, we use the available MeCab (mecab- python3 1.0.3), Sudachi (SudachiPy 0.5.1), and SentencePiece (sentencepiece 0.1.94) packages available via the Python Package Index (PyPI)[7]. As for the TF- IDF vectorizer, Multinomial Naïve Bayes classifier, and Logistic Regression classifier, we used packages provided by Scikit-learn[8].

2.1.4 Methods

After preparing our environment specifications as previously described, we could then proceed to conduct our experiments. The overall flow of our experiments can be seen in Fig. 1. After the original Rakuten binary dataset is downloaded, firstly, we randomly sampled ten percent of the provided train and test data, resulting in a total of 340,000 train data and 40,000 test data. We then proceed to train a SentencePiece (SP) model based on the sampled train data, setting the vocab_size parameter as 32,000. We initially chose 32,000 based on the result shown in experiments in a previous work [23].

Unlike SentencePiece, Sudachi has its own dictionary with different sizes and MeCab can be integrated with various existing dictionaries in order to perform tokenization, so there is no need for more pre-training. We use Sudachi's core dictionary for surface-form tokenization using Sudachi, and we use the unidic-lite dictionary for tokenization with MeCab. Furthermore, before building our classification model, we experimented with few randomly selected reviews to get a glimpse of the tokenization results by the three different tokenizers.

After that, using each tokenizer, we vectorize all the reviews in the training set to create matrices of TF-IDF values which will then be used to build our classification model. In this process, we also calculated the time spent by each tokenizer to process various number of train data into their TF-IDF

[7] https://pypi.org

[8] https://scikit-learn.org/stable/

values. We then use the vectorized data to build two classification models for each tokenizer, one using the Multinomial Naïve Bayes (MNB) classifier, and another one using the Logistic Regression (LR) classifier. Finally, we test our models to classify reviews in the test set and evaluate the error percentages.

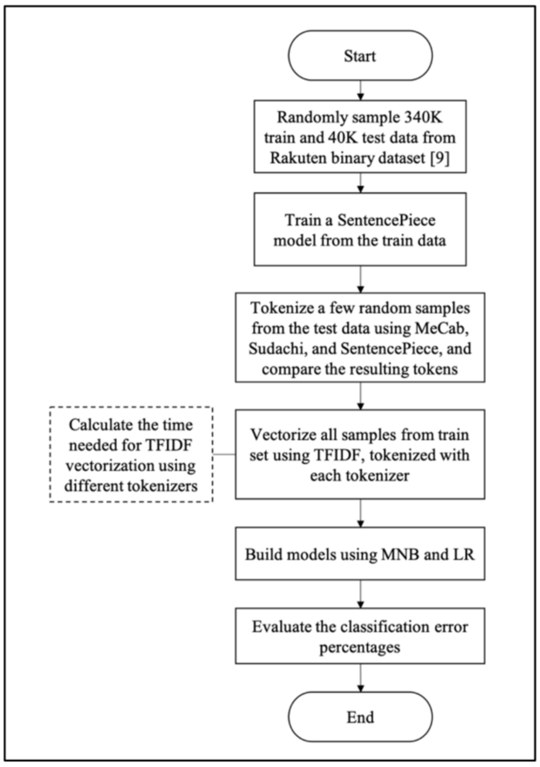

Start
Randomly sample 340K train and 40K test data from Rakuten binary dataset [9]
Train a SentencePiece model from the train data
Tokenize a few random samples from the test data using MeCab, Sudachi, and SentencePiece, and compare the resulting tokens
Calculate the time needed for TFIDF vectorization using different tokenizers
Vectorize all samples from train set using TFIDF, tokenized with each tokenizer
Build models using MNB and LR
Evaluate the classification error percentages
End

Fig. 1 Flowchart of the experiment

2.1.5 Tokenization Result

We selected several review texts from the train and test set, then try to segment the words or subwords in order to experiment with each tokenizer. Vocab_size=32,000 is used for the SentencePiece model. Some parts of review texts along with their tokenization results are as can be seen in Table 1.

Table 1 Examples of tokenization result using each tokenizer

1	
Original	“自転車通勤用に購入。サックス を選びましたが、...”
MeCab	['自転', '車', '通勤', '用', 'に', '購入', '。', 'サックス', 'を', '選び', 'まし', 'た', 'が', ...]
Sudachi	['自転車', '通勤用', 'に', '購入', '。', 'サックス', 'を', '選び', 'まし', 'た', 'が', ...]
SentencePiece	['▁', '自転車通勤', '用に購入', '。', 'サックス', 'を選びましたが', ...]
2	
Original	“かわいいです(*^。^*)¥¥n パソコン...”
MeCab	['かわいい', 'です', '(', '*^。^*)¥¥', 'n', 'パソコン', ...]

Sudachi	['かわいい', 'です', '(*^。^*)', '¥¥', 'n', 'パソコン', ...]
SentencePiece	['▁', 'かわいいです', '(*^。^*)', '¥¥', 'n', 'パソコン', ...]

The examples provided above are randomly selected and might not be fully representative to showcase the full capabilities of each tokenizer, however, some general similarities and differences can be observed. Compared to MeCab and Sudachi, tokens generated by SentencePiece are not based on any formal dictionary, so there are words or subwords that does not match formal Japanese dictionary, for example the token " 便利ですね " ("benridesune") which is usually segmented into three words ("benri", "desu", and "ne") but treated as one token. Moreover, although tokens generated by MeCab and Sudachi are generally similar, there are some characters that are treated different depending on the context they are in. For example, the words "自転車通勤用" and the combination of characters comprising the emoji "(*^。^*)", are segmented differently in MeCab and Sudachi. Sudachi could divide the characters into ["自転車"(bicycle), "通勤用"(for commuting)] and treat the whole emoji as one union, while MeCab further separate the combination into ["自転", "車", "通勤", "用"] and ["(", "*^。^*)¥¥"].

2.1.6 Elapsed Time for Vectorization Using TF-IDF

We then proceed to use each tokenizer to vectorize the all texts in our train data (340,000 review texts) using TF- IDF and calculate the time elapsed. As can be seen in Table 2, SentencePiece is the fastest while MeCab is only slightly slower, and Sudachi needs the longest time. This might be caused by Sudachi's focus in providing high quality segmentation based on its continuously updated rules and dictionary that enable it to divide words as can be seen in the previous paragraph.

Table 2 Elapsed time to vectorize TF-IDF values

Tokenizer	**Elapsed Time (seconds)**
MeCab	34.65
Sudachi	1533.58
SentencePiece	25.84

2.1.7 Text Classification Model Performance

After observing the tokens generated by each tokenizer and calculated the time elapsed to vectorize TF-IDF values, we proceed to use the TF-IDF values generated by each tokenizer and train two models using Logistic Regression (LR) and Multinomial Naïve Bayes (MNB). We could observe the effect of each tokenizer, with varying tokenization approaches and time elapsed to generate the TF-IDF values, on the classification performances in Table 3.

In our experiment using linear model such as Logistic Regression, and also Multinomial Naïve Bayes, with TF- IDF vectorizer, the combination of SentencePiece with Logistic Regression outperforms the others with 6.54 error percentage on the training set and 8.02 error percentage on the testing set. We can see that even though the segmentation results by SentencePiece are quite different with the other tokenizers' results, it seems to work well on a linear classifier to solve binary classification problem. Another finding is that in our task, even though Sudachi, with its 'surface form' tokenization, could perform relatively better word segmentation, the resulting classification model could perform only slightly better than MeCab, despite taking the longest elapsed time in the vectorization phase.

Table 3 Classification error percentages

Tokenizer	**Classifier**	**Error Train (340,000)**	**Error Test (40,000)**
MeCab	Logistic Regression	8.52	9.63
Sudachi	Logistic Regression	8.5	9.59
SentencePiece	Logistic Regression	6.54	8.02
MeCab	Multinomial Naïve Bayes	11.24	12.52
Sudachi	Multinomial Naïve Bayes	11.04	12.42
SentencePiece	Multinomial Naïve Bayes	8.28	8.9

Furthermore, using the best performing model in Table 2 (Logistic Regression with SentencePiece), we then perform a hyperparameter tuning process for Logistic Regression using grid search and repeated stratified k-fold cross validator from Scikit-learn. Table 3 shows the error percentages of our final model (train error: 6.54, test error: 8.02) using the following hyperparameters on the Logistic Regression classifier: C=10, penalty='l2', solver='lbfgs'.

Table 4 Error percentages after hyperparameter tuning

Tokenizer	**Classifier**	**Error Train**	**Error Test**
SentencePiece	Logistic Regression	5.56	7.78

2.1.8 Conclusion

This experiment reports the results of an evaluation of three popular tokenization tools, MeCab, Sudachi, and SentencePiece, for processing Japanese texts. The resulting tokens are then used to train text classification models using TF-IDF with Logistic Regression and Multinomial Naïve Bayes. We found that the generated tokens from Sudachi are more likely to match dictionary results and common words understood by human, however, MeCab and SentencePiece are significantly faster. Moreover, even though tokens generated by SentencePiece are limited to its training data and might not match common dictionary results, they perform better for our dataset, which is a binary sentiment-based text classification task. Finally, the combination of SentencePiece, TF-IDF, and Logistic Regression achieved the best performance with 5.56 training error percentage and 7.78 testing error percentage.

2.2 Cross-Lingual Large Language Models (LLMs)

Based on the experiment in the previous section, depending on the use case, SentencePiece can be a great choice of tool for tokenizing Japanese texts. Additionally, since it is a language-agnostic tokenizer, it can also be used for various other languages. In this experiment, we further explore the applicability of SentencePiece, along with increasingly large and inclusive Transformers-based language models in processing cross-lingual texts, that include Japanese. The reason being, while high-resource languages have achieved great successes in various tasks, other languages with limited data and computational resources are still left behind. This poses a challenge for researchers and practitioners in foreign language learning field, because even though many language models are publicly available for English, it is not the case for other languages which could be the native language of the learners. For example, it may not be that hard for Spanish speakers to understand English texts since the accuracy of machine translation models for Spanish to English is relatively good, one of the reasons is that the resources to build performant language models for both languages are relatively abundant, compared to, for example, speakers of the Sundanese language (a local language in Indonesia) who are trying to understand English texts.

Cross-lingual transfer learning, where a high-resource language is used to train a downstream task model to improve the model's performance in another target language, shows a promising potential to tackle this setback. Many works of research have experimented and shown the potential benefit of using cross-lingual transfer in several NLP tasks, such as machine translation [29] [30], and named entity recognition [31] [32]. Recently, cross-lingual transfer learning has become an essential tool for improving performance in various downstream NLP tasks. This is also thanks to the recent advancements of massively multi-lingual Transformers pre-trained models, including mBERT [10], XLM [33], and the most recent one being XLM-RoBERTa (XLM-R) [34].

Even though the multi-lingual field has grown a lot, there are still some challenges. For example, previous works have reported that the quality of unsupervised cross-lingual word embedding is susceptible to the choice of language pairs and the comparability of the monolingual data [35], thus limiting the performance when the source and target language have different linguistic structures (e.g., English and Japanese) [36]. XLM-R is trained using 100 languages globally and achieved SOTA results in various multi-lingual NLP tasks such as the XNLI, Named Entity Recognition, Cross-lingual question answering, and the GLUE benchmark [34]. However, it reports only the performance evaluation for some of the languages used during pretraining such as English, French, Swahili, and Urdu.

This experiment explores the XLM-R base pre-trained model's capability for cross-lingual transfer learning encompassing English, Japanese, and Indonesian. We compare the performance with models from previous works, evaluate zero-shot transfer learning capability, and fine-tune mono- and multi-lingual models in a supervised manner for comparison purposes. Results show that both fine-tuning and zero-shot transfer learning from English to Japanese and Indonesian for a downstream task; in this case, binary sentiment classification, using XLM-R, yields promising results. All experiments are conducted using Google Colaboratory. We also report the full specifications and hyperparameters used in our experiments for reproducibility and provide an overview of the applicability of using XLM-R for zero- shot transfer learning within a limited amount of data and computational resources.

2.2.1 Related Works

As a massively multi-lingual Transformers (MMT) model, XLM-R [34] is a robustly trained RoBERTa, exposed to a much larger multi-lingual corpus than mBERT. It is trained on the CommonCrawl-100 data of 100 languages. There are 88 languages in the intersection of XLM-R's and mBERT's corpora; for some languages (e.g., Kiswahili), XLM-R's monolingual data are several orders of magnitude larger than with mBERT. There are many methods for performing cross-lingual transfer based on MMTs, some of which are fine-tuning [37] and zero- shot transfer learning [38] [32]. The common thread is that data in a high-resource source language can be used to improve performance on a low-resource target language.

Even though XLM-R is pre-trained using 100 languages, investigations regarding the applicability of XLM-R for downstream tasks in some languages with less resource than English, such as Japanese and Indonesian, with thorough experiments and reproducible results are still limited. Several works have tried to implement cross- lingual transfer learning using several Japanese and Indonesian text classification models. For Japanese, previous works have shown the capability of XLM-R for Japanese for dependency parsing [39] and named entity recognition [40], but no thorough comparison for cross-lingual transfer learning (fine-tuned and zero-shot) for sentiment classification are provided. In the Multi-lingual Amazon Review Corpus [41] in which Japanese is one of the languages of the corpus, the authors provided baseline sentiment classification performance using mBERT, but performance using XLM- R has not been reported. For Indonesian, previous works have shown the capability of using BERT and XLM-R for various tasks including sentiment classification [42] [43], however, the results are mainly focused on building powerful monolingual models for Indonesian.

Furthermore, unlike English, Japanese texts contain no whitespace and there are various ways to split sentences into words, with each split could end in a different meaning and nuance. In order to tackle this problem, a recent work proposed a language-independent subword tokenizer and detokenizer designed for neural-based text processing, named SentencePiece [21]. Its performance is shown to be effective for various tasks involving language pairs with different character sets, such as English- Japanese neural machine translation [21] and sentiment analysis in Japanese [22]. It is also utilized by state-of- the-art cross-lingual models such as XLM-R, which is the focus of our current research.

2.2.2 Dataset

We gathered binary sentiment datasets from several sources and put shorthand nicknames on each dataset to be addressed in the following sections.

1. *AmazonEN*: English Amazon product review sentiment dataset from The Multi-lingual Amazon Reviews Corpus [41]. We use 160,000 data for fine- tuning. 4,000 data for evaluation.
2. *AmazonJA*: Japanese Amazon product review sentiment dataset from The Multi-lingual Amazon Reviews Corpus [41]. We use 160,000 data for fine- tuning. 4,000 data for evaluation.
3. *RakutenJA*: Japanese Rakuten product review binary sentiment dataset from [26]. We use 400,000 data for evaluation.
4. *IndolemID*: Indonesian Twitter and hotel review sentiment dataset from IndoLEM dataset [43]. We use 5,048 data for evaluation.
5. *SmsaID*: Indonesian multi-platform review sentiment dataset from SmSA dataset [44]. We use 1,129 data for evaluation.

For each dataset, we use the review body/text as the input and the sentiment (0 for negative and 1 for positive) as the classification label.

2.2.3 Experimental Setup

In this experiment, we use the free version of Google Colab with GPU for all our experiments. Due to the dynamic GPU allocation by Google Colab, two GPU types are used in our experiment: Tesla T4 and Tesla P100-PCIE-16GB. Review sentiments gathered from *AmazonEN* and *AmazonJA* are rated as a 5-star rating. Following the original paper's practice [41], we converted the 1- and 2-stars rating as negative the 4- and 5-stars rating as positive reviews, we omit the 3-stars rating. Indonesian review data gathered from *SmsaID* initially contains three classes, which are positive, negative, and neutral; similarly, we omit the neutral class in this experiment.

Furthermore, to provide an equal comparison of our models' performance, we use similar metrics used in the sources of each dataset. Specifically, we use the error percentage for *AmazonEN*, *AmazonJA*, *RakutenJA*, and the macro-averaged F1-score for *SmsaID* and *IndolemID*. Additionally, we report the hyper-parameters and the time needed for fine-tuning the models to provide a general overview of the applicability and resource needed by the readers to reproduce the results of our experiments. We divide the experiments into two scenarios:

- Fine-tuned supervised learning
  Fine-tune the XLM-RoBERTa$_{\text{BASE}}$ pre-trained model using *AmazonEN*, *AmazonJA*, and the combination of English and Japanese Amazon reviews (*AmazonENJA*), then evaluate the models in monolingual and multi-lingual settings.
- Zero-shot transfer learning
  Use the fine-tuned model using *AmazonEN* to evaluate zero-shot cross-lingual transfer learning capability in *AmazonJA*, *RakutenJA*, *SmsaID*, and *IndolemID* datasets.

2.2.4 Results: Fine-Tuned Supervised Learning and Zero-shot Transfer Learning

Firstly, this section reports the result of fine-tuning the XLM-R model. We fine-tuned the xlm-roberta-base pre-trained model from the HuggingFace transformers library, three times, using

AmazonEN, *AmazonJA*, and *AmazonENJA*. For *AmazonEN* and *AmazonJA*, our final models are fine-tuned using the linear scheduler with warmup, 4 epochs, batch size=32, optimizer=AdamW, and learning rate=2e- 5. For *AmazonENJA*, the final model uses the same above parameters but only with 2 epochs. Table 5 shows the GPU and averaged elapsed time for each epoch in the fine-tuning process.

Table 5 Specification and elapsed time for fine-tuning XLM-R

Training Data	**GPU**	**No. of Epoch(s)**	**Avg. Elapsed Time per Epoch**
AmazonEN	Tesla T4	4	33 minutes 5 seconds
AmazonJA	Tesla P100-PCIE-16GB	4	17 minutes 31 seconds
AmazonENJA	Tesla P100-PCIE-16GB	2	35 minutes 57 seconds

After fine-tuning three models in the previous step, we evaluate each of the fine-tuned models for supervised learning performance on the exact language from which the model is fine-tuned. We calculate the error percentage for the model prediction on test data and compare the results with a baseline model trained using mBERT, as displayed in Table 6. It can be seen that XLM-R outperforms mBERT in all three supervised models for binary sentiment classification in English and Japanese. There is no comparison from the baseline model for the multi-lingual model fine-tuned using *AmazonEN* and *AmazonJA*. However, it can be seen that it is possible to have a single bi-lingual model for both languages, eliminating the need to set up multiple models for every language.

Table 6 Error percentage of the fully-supervised evaluation on the Multi-lingual Amazon Review Corpus. Results using mBERT are obtained from *[41]*.

Model	**EN-only**	**JA-only**	**EN&JA**
mBERT	8.8	11.1	-
XLM-R$_{\text{BASE}}$	**7.35**	**7.25**	**7.19**

Secondly, we use the fine-tuned models from the previous scenario to evaluate the applicability of zero- shot cross-lingual transfer learning from one language to the others. Table 7 shows the results (error percentage, lower is better) of the experiments conducted in this scenario using the multi-lingual Amazon and Japanese Rakuten data. Additionally, Table 8 reports the results (F-1 score, higher is better) using the multi-platform Indonesian sentiment datasets.

Table 7 Error percentage of zero-shot cross-lingual transfer learning using XLM-R$_{\text{BASE}}$ in comparison to a zero-shot mBERT from English data *[41]* and Japanese data *[22]*

Model	**AmazonJA**	**RakutenJA**
Zero-shot mBERT	19.04	-
Fully-supervised ULMFiT	-	**4.45**
XLM-R$_{\text{BASE}}$ w/ *AmazonEN*	11.12	13.09

XLM-R$_{\text{BASE}}$ w/ *AmazonENJA*	**7.05**	8.51

Table 8 Macro-averaged F1-score of zero-shot cross-lingual transfer learning using XLM-R$_{\text{BASE}}$ for Indonesian

Model	**IndolemID**	**SmsaID**
Fully-supervised BERT	**84.13**	**92.72**
Fully-supervised mBERT	76.58	84.14
XLM-R$_{\text{BASE}}$ w/ *AmazonEN*	72.19	86.77
XLM-R$_{\text{BASE}}$ w/ *AmazonENJA*	73.31	87.99

On the Japanese product review from the Amazon dataset, in Table 3, our model achieves a better error percentage of 11.12. It is almost 8 points better than the original baseline model, which is also evaluated using zero-shot cross-lingual transfer learning from English to Japanese, using mBERT. On another Japanese dataset with more evaluation data (400,000 reviews), the Rakuten dataset, our model achieves a 13.09 error percentage with zero-shot from English. Furthermore, it achieves a much better score of 8.51 error percentage if we add a substantial 160,000 review data from *AmazonJA* when fine-tuning the model. Although they are from different platforms, product review data shares similar patterns. This result is still far from the SOTA result of the 4.45 error percentage achieved by previous work, in which the model is trained in a fully-supervised monolingual setting using BERT.

For the Indonesian sentiment datasets, as also described in Table 8, we use two datasets with comparable results from different sources for evaluation purposes. Macro- averaged F1 score is used to evaluate the Indonesian datasets to follow previous works for comparison purposes. We experimented with zero-shot cross-lingual transfer learning using two models for classifying the Indonesian datasets; one is trained only with 160,000 English Amazon reviews, and another one containing English and Japanese Amazon reviews. In both Indonesian datasets, we can see a pattern of more data leads to better performance. Similar to prior research results [34], the model trained with multilingual data, in our case, Japanese and English review data, performs better. The XLM-R$_{\text{BASE}}$ w/ *AmazonENJA* model achieves a 73.31 F1-score on the *IndolemID* dataset, outperforming a previous model, trained with mBERT in a fully- supervised monolingual setting. Moreover, the same model achieves a better F1-score of 88 on the *SmsaID* dataset, outperforming another mBERT model trained in a fully-supervised monolingual setting. For both datasets, our model performance is still worse when compared to the SOTA model's result, as shown in Table 7 and Table 8.

Based on the results above, it is essential to note that the models used to compare *RakutenJA*, *IndolemID*, and *SmsaID* are trained in a fully-supervised approach using the same language with the target language. In contrast, our model, which is trained using *AmazonEN*, has never seen Japanese product reviews, and our models trained with *AmazonEN* and *AmazonENJA* have never seen

Indonesian review texts. Furthermore, the models compared in the previous tables are trained mainly by much bigger architectures with more epochs, which means training them will need much more computational resources and time. Our current experiments show the applicability of zero-shot cross-lingual transfer learning with less computational costs.

2.2.5 Conclusion

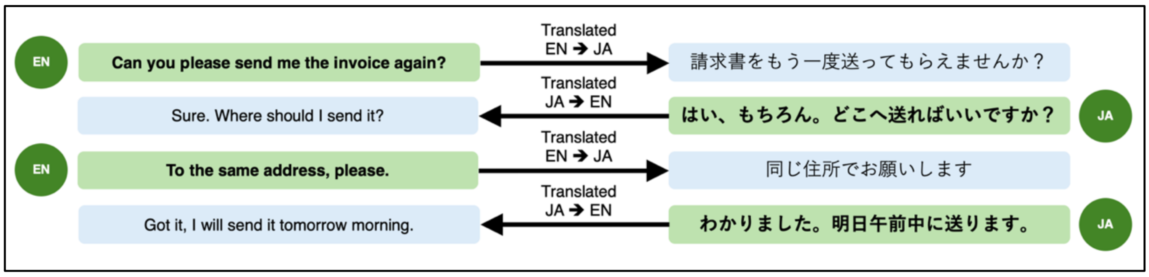
This section reports the results of experiments focusing on evaluating the applicability of cross-lingual transfer learning using the XLM-R pre-trained model. We then compare the results with previous works to provide an overview of the zero-shot approach's capability using XLM-R to use a multi-lingual model for bilingual data instead of one model for one language. Based on the results, zero-shot cross-lingual transfer learning yields promising results using XLM-R. All experiments are performed using the free version of Google Colab. The models achieve the best result in one dataset and shows the applicability of cross-lingual transfer learning, considering that the models have not seen languages in the target dataset, it can outperform SOTA results in other datasets trained in a fully supervised approach.

Chapter III Japanese-English Neural Machine Translation and Its Challenges

3.1 Introduction to Japanese-English NMT and Anaphoric Zero-Pronoun

In recent years, many studies in the deep learning and natural language processing fields have progressed toward enabling the building of high-performance neural machine translation (NMT) models, most notably the neural encoder-decoder models with attention [45] [3]. NMTs are a popular research field that is being applied in various forms of applied technologies to assist human communication. Many studies have achieved various translation milestones in various language directions, such as translation in low-resource languages [46] [47] [48]. In some language directions and domains, sentence-level translation has been considered similar to human translation [49]. However, document-level translation remains a challenging field, particularly in the context of chat translation, in which sentences are typically spoken language instead of written language, and contain many references to previous sentences.

Fig. 2 An example of multilingual chat conversation supported by machine translation


EN
Can you please send me the invoice again?
Translated EN → JA
請求書をもう一度送ってもらえませんか？
Translated JA → EN
Sure. Where should I send it?
はい、もちろん。どこへ送ればいいですか？
JA
EN
To the same address, please.
Translated EN → JA
同じ住所でお願いします
Translated JA → EN
Got it, I will send it tomorrow morning.
わかりました。明日午前中に送ります。
JA

As technology continues advancing, communicating with people in different languages via chat messages has become a trivial task. Various companies worldwide are already providing multilingual chat translation services and products [50] [51] [52], as Fig. 2 illustrates. A challenge in translating chat messages is the anaphoric zero-pronoun phenomenon, in which pronouns are dropped or omitted in a sentence. This arises from pronouns such as subjects, objects, and possessive cases are often omitted from conversational sentences. The sentences in Table 9 were taken from the business scene dialog corpus [53], which contains parallel conversations in Japanese and English. This anaphoric zero-pronoun language is nonexclusive to Japanese. However, the surrounding words in a zero-pronoun Japanese sentence typically do not have inflectional forms that change depending on the previously omitted pronoun [54], which makes the Japanese language one of the most difficult languages to resolve [55] [54].

The main contribution of this section is to provide a support tool for building zero-pronoun evaluation sets, namely 0Past, to assist in the evaluation of NMT models translate heavy pro-drop languages, such as Japanese. This section also reports two additional findings: (1) performance comparison, in terms of BLEU score, between three NMT models to translate chat messages from Japanese to English demonstrating that specific evaluation sets for zero-pronoun sentences are required; and (2) two zero-pronoun evaluation sets labeled using the proposed support tool.

Table 9 Example of conversation containing sentences with omitted pronouns

Speaker	Language	Sentence
Speaker #1	Japanese	私がそっちに行ってもいいと思っていたんですけど。
	English	**I** was thinking I can go to **your** place though.
Speaker #2	Japanese	大丈夫ですよ、近いですし。
	English	It's okay, **it's** close.
Speaker #1	Japanese	じゃあ、明日の１０時半ごろ、お伺いしますね。
	English	Okay, **I** will go to **your** place around ten thirty tomorrow.
Speaker #2	Japanese	はい、わかりました。
	English	Okay, got it

3.2 Related Works

3.2.1 Japanese-English Neural Machine Translation with Transformers

Progress in NMT has been rapidly advancing in recent years. The machines' ability to translate certain language pairs at the sentence level, ignoring its contexts, is considered to perform competitively with human translation. Most state-of-the-art NMT models are based on the encoder-decoder architecture [45]in which the encoder layers map the source sentence into word vectors, and the decoder layers produce the target sentence given the source sentences represented by its word vectors. This approach assumes a conditional independence of each sentence and then uses it to translate sentences from the source to the target language, ignoring both source- and target-side context sentences. As such, these systems optimize the negative log-likelihood of the sentences:

$$p\left(y^{(k)}\middle|x^{(k)}\right) = \prod_{t=1}^{n}\left(y_t^{(k)}\middle|y_{<t}^{(k)}, x^{(k)}\right) \tag{1}$$

where $x^{(k)}$ and $y^{(k)}$ are the k-th source and target training sentences, respectively, and $y_t^{(k)}$is the t-th token in $y^{(k)}$ (Lopes, et al., 2020).

Several studies have proposed the use of context-aware NMT approaches that integrate source- and target-side contexts [4] [26] [6]. Existing studies have shown that human translators clearly outperform NMT models when the translation of the target language must consider the contexts at the document level in Chinese-English translations [56] [57]. A previous study [58] conducted a systematic comparison of context-aware NMT methods using large datasets that included findings for anaphoric pronoun translation. However, it focused only on three language directions: English to French, German, and Brazilian Portuguese. Studies related to the evaluation and improvement of discourse translation in Japanese, which has their own challenges, still lack comparisons to other languages.

3.2.2 Corpora for Discourse Translation in Japanese

Many of the publicly available parallel corpora for training a Japanese-English NMT model are constructed with sentences from written language, such as crawled documents from the web, patents [59], and scientific papers [60]. Only in recent years has more work started to shift focus to providing corpora for discourse translation in Japanese. One research paper [53] introduced a Japanese-English business conversation parallel corpus, the Business Scene Dialog (BSD) corpus. This corpus contains 955 scenarios with 30,000 parallel sentences. It provides a publicly available parallel corpus for training and evaluating conversational NMT for Japanese to English but lacks an evaluation set for measuring the models' capability to handle language-specific phenomena, such as anaphoric zero resolution.

Another study [61] separately constructed a test set for evaluating zero-pronoun resolution in Japanese-English translations, which consists of four components: a current sentence, context, a correct translation, and an incorrect translation. This type of test set is particularly useful because it contains zero pronoun information; therefore, translation models must predict the correct antecedent based on the contexts in previous sentences inside one conversation. However, manually handcrafting them requires time and effort; thus, such sets are rare, particularly those publicly available. Our research aims to kickstart the development of support tools that can handle specific language phenomena to assist the evaluation of model performance in discourse translation and, ultimately, improve the overall capability of machine translation models.

3.2.3 Support Tool for Building Evaluation Sets

Human involvement in various machine translation (MT) tasks, such as the creation and annotation of parallel corpora and manual evaluation, is considered important, although time-consuming and non-trivial. This is challenging for many reasons, one of which is that translation results produced by an NMT model are not guaranteed to be syntactically well-formed and easy to understand; thus, identifying the exact differences between translation candidates is challenging, particularly for a long or conversational document that contains many coreferences between sentences. Recently, the 2020 Conference on Machine Translation (WMT '20) developed a human intelligence task (HIT) tool [62] based on a previously developed Appraise toolkit [63] to evaluate a shared task in chat translation. This tool provides a UI specifically designed to facilitate translation evaluation for the human evaluator in assessing the WMT '20 task submissions; however, although the tool provides modern UI for easier use, it focuses only on the human evaluation of machine translations, which are then compared to its machine counterparts to score the overall translation results without information regarding the model's capability in handling language-specific phenomena, such as anaphoric zero pronouns.

However, previous studies have proposed various tools to assist data annotation activities. WebAnno [64] presented a general-purpose web-based annotation tool for a wide range of linguistic annotations, such as parts-of-speech called entities, dependency parsing, and co-reference chain annotations. In another study, INCEpTION [65] provided an annotation platform for tasks, including interactive and semantic annotation. It is built based on the shortcomings of previous support tools, including WebAnno's split-screen mode, which was considered tedious for users. Such tools provide a wide range of features to support manual annotation, partially supported by pre-annotation features that leverage machine-learning methodologies. However, these tools were built for traditional NLP tasks tailored for document-like texts, such as parts-of-speech tagging and entity linking recommendations.

Our proposed tool, the Zero-Pronoun Annotation Support Tool (0Past), was designed and built to handle chat annotation and evaluation. For example, it outlines the boundaries between each speech of a certain speaker, facilitating navigation between conversations. The tool focuses on the anaphoric zero-pronoun phenomenon, which is a challenging task in chat translation on several languages, particularly in Japanese. The proposed tool is built on existing tools, tailored to support the annotation and evaluation of chat messages, to improve the performance of NMT models in handling sentences with anaphoric zero-pronouns. The following sections describe the complete list of current and potential features of 0Past.

3.3 Proposed Tool: Zero-Pronoun Annotation Support Tool (0Past)

3.3.1 Features Overview

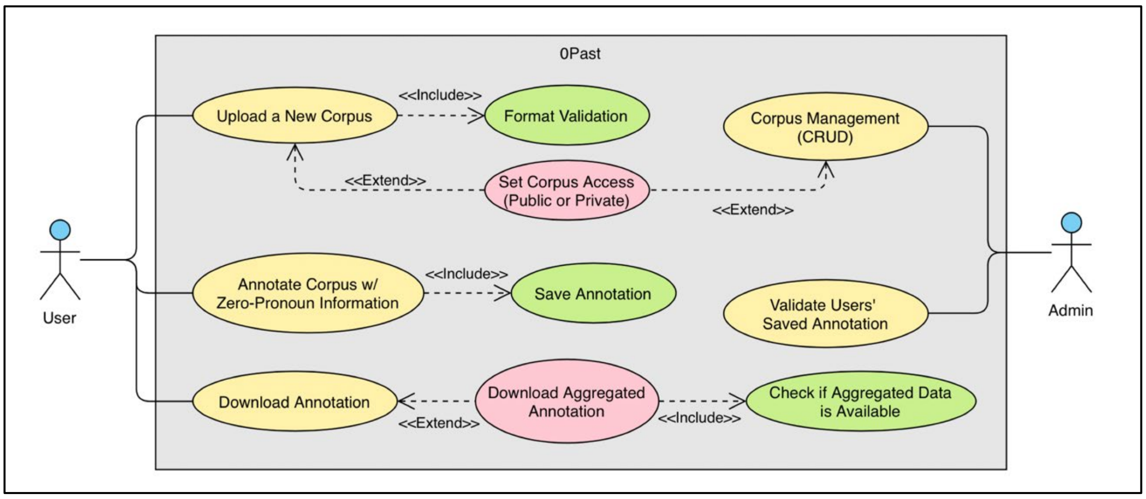

0Past
User
Admin
Upload a New Corpus
<<Include>>
Format Validation
Corpus Management (CRUD)
<<Extend>>
Set Corpus Access (Public or Private)
<<Extend>>
Annotate Corpus w/ Zero-Pronoun Information
<<Include>>
Save Annotation
Validate Users' Saved Annotation
Download Annotation
<<Extend>>
Download Aggregated Annotation
<<Include>>
Check if Aggregated Data is Available

Fig. 3 Use case diagram of 0Past

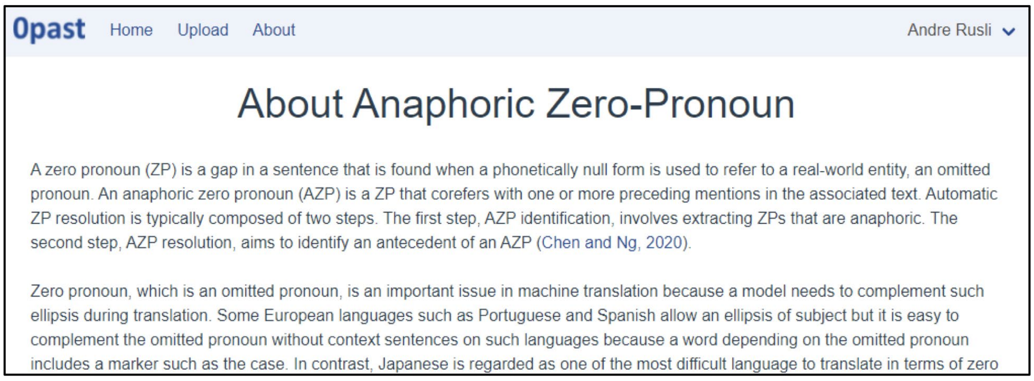
The proposed tool was developed using VueJS[9] and Buefy[10] as the front-end frameworks and the Firebase Realtime Database[11] to store annotations. Fig. 3 shows a use-case diagram of the proposed tool with two types of users: a regular user (User) and an administrator (Admin). Regular users can perform three actions: upload a new parallel corpus, annotating an uploaded corpus, and download the annotation data. In turn, the administrator can perform two actions: manage the uploaded

[9] http://vuejs.org

[10] http://buefy.org

[11] https://firebase.google.com/docs/database

corpus inside the application and validate annotations made by the users, which can later be used to create the aggregated annotation of a certain dataset. Fig. 4 shows the About page, which explains the anaphoric zero-pronoun with three navigation items: Home, Upload, and About. Home is the page where users can select a corpus to annotate and download its annotation data, whereas Upload provides the interface to upload new corpora. The following paragraphs explain these features in detail.



About Anaphoric Zero-Pronoun

A zero pronoun (ZP) is a gap in a sentence that is found when a phonetically null form is used to refer to a real-world entity, an omitted pronoun. An anaphoric zero pronoun (AZP) is a ZP that corefers with one or more preceding mentions in the associated text. Automatic ZP resolution is typically composed of two steps. The first step, AZP identification, involves extracting ZPs that are anaphoric. The second step, AZP resolution, aims to identify an antecedent of an AZP (Chen and Ng, 2020).

Zero pronoun, which is an omitted pronoun, is an important issue in machine translation because a model needs to complement such ellipsis during translation. Some European languages such as Portuguese and Spanish allow an ellipsis of subject but it is easy to complement the omitted pronoun without context sentences on such languages because a word depending on the omitted pronoun includes a marker such as the case. In contrast, Japanese is regarded as one of the most difficult language to translate in terms of zero

Fig. 4 About anaphoric zero-pronoun in 0Past

The first feature allows users to upload parallel corpora that are currently unavailable in the system. A specific format is available for download that the corpus must follow. Thus, users must follow the correct format, and the system checks this before enabling the upload. The format used in the current version of 0Past follows the JSON format, as shown in Table 10. By default, the uploaded corpus can only be accessed by the uploader. However, users can also choose whether the uploaded corpus can be accessed publicly. If a corpus is set as public, then other users of 0Past can access it and provide their annotations, which are saved in each of the user’s account.

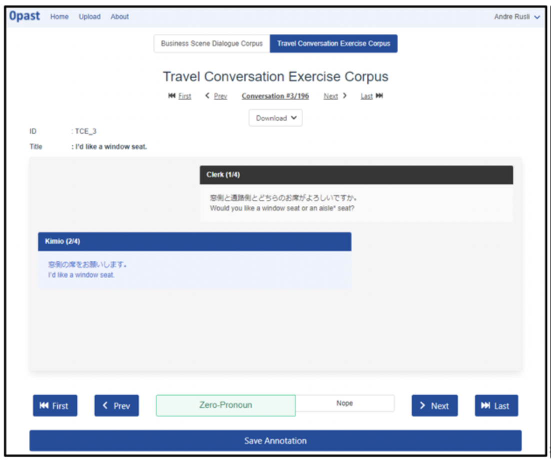
Next is the primary feature of 0Past, which is users viewing uploaded parallel corpora and contributing to the community by providing their zero-pronoun annotations for the available corpora. Fig. 5 depicts the main user interface for annotation, where users can navigate between sentences and conversations within a dataset. The proposed tool provides a chat-like user interface and displays each conversation separately. While the user views a certain conversation, the messages within the conversation are displayed independently with a distinct color for the newest message. Users can see messages back and forth in the conversation with the speaker’s name displayed above along with the chat number, allowing them to have a sense of direction in terms of where they are in the conversation. Users can then annotate each sentence by selecting whether the zero-pronoun phenomenon exists in the corresponding sentence.

Table 10 Content format inside a corpus file to be uploaded to 0Past

JSON Format	**Example (data taken from BSD Dev Set)**

```
[
    {
        "id": string,
        "tag":   string,
        "title":   string,
        "original_language":   string,
        "conversation": [
            {
                "no": number,
                "en_speaker":   string,
                "ja_speaker":   string,
                "en_sentence":   string,
                "ja_sentence":   string,
            },
        ]
    },
]
```

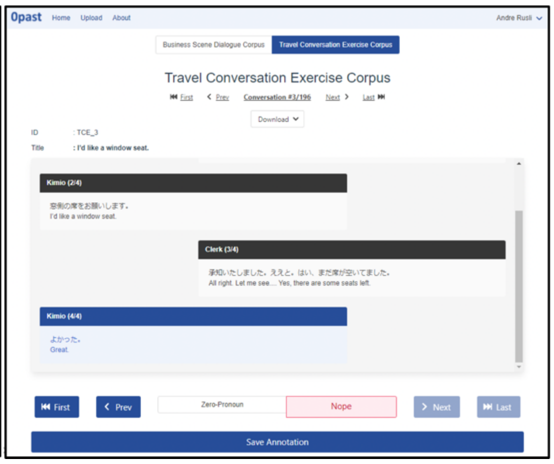
```
[
    {
        "id": "190315_E001_17",
        "tag": "training",
        "title": "Training: How to do research",
        "original_language": "en",
        "conversation": [
            {
                "no": 1,
                "en_speaker": "Mr. Ben Sherman",
                "ja_speaker": "ベン シャーマンさん",
                "en_sentence": "I will be teaching you how to conduct research today.",
                "ja_sentence": "今日は調査の進め方についてトレーニングします。"
            },
        ]
    },
]
```



ID : TCE_3
Title : I'd like a window seat.

Clerk (1/4)
窓側と通路側とどちらのお席がよろしいですか。
Would you like a window seat or an aisle* seat?

Kimio (2/4)
窓側の席をお願いします。
I'd like a window seat.



ID : TCE_3
Title : I'd like a window seat.

Kimio (2/4)
窓側の席をお願いします。
I'd like a window seat.

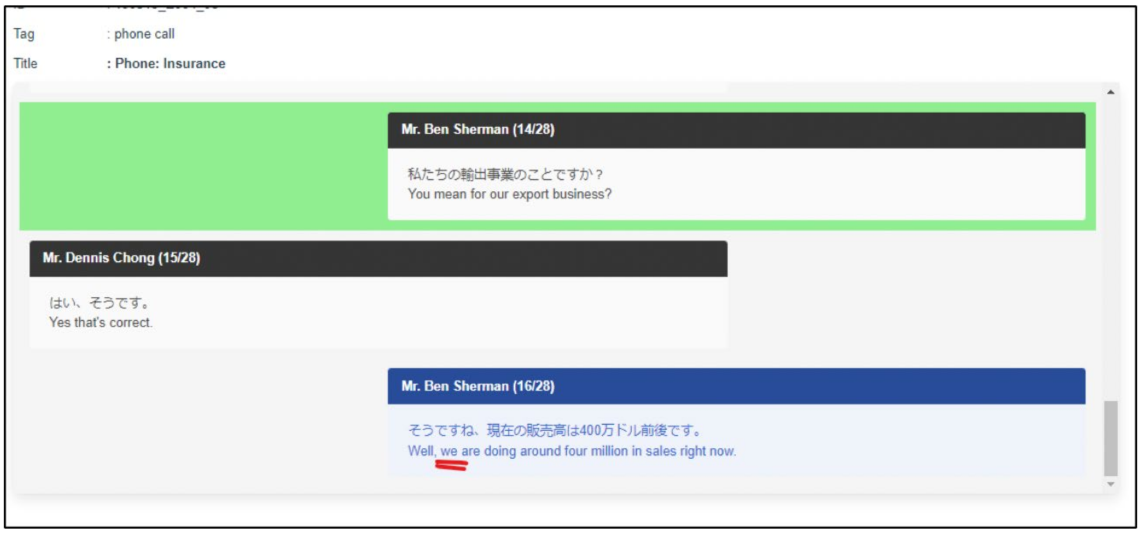
Clerk (3/4)
承知いたしました。ええと。はい、まだ席が空いてました。
All right. Let me see.... Yes, there are some seats left.

Kimio (4/4)
よかった。
Great.

Fig. 5 Sentences and their annotations from the TCE corpus displayed in 0Past

Furthermore, in addition to labeling each sentence as a zero-pronoun or not, users can also annotate which of the previous sentences, if any, act as context sentences that decide the content of the omitted pronoun. As shown in Fig. 6, the current sentence is “そうですね、現在の販売高は 400

万ドル前後です。” and is labeled a zero-pronoun sentence. In English, this sentence is “Well, we are doing around four million in sales right now.” The omitted pronoun is “we,” however, knowing this by looking at this sentence only is difficult, even for humans. However, if we look at the previous sentences, some sentences provide contexts, thereby helping us decide that the omitted pronoun is indeed “we.” To provide this type of annotation, users can scroll to see the previous sentences and click on the previous sentences that provide the context. The clicked sentence is then colored green, as shown in Fig. 6.


Tag : phone call
Title : Phone: Insurance

Mr. Ben Sherman (14/28)
私たちの輸出事業のことですか？
You mean for our export business?

Mr. Dennis Chong (15/28)
はい、そうです。
Yes that's correct.

Mr. Ben Sherman (16/28)
そうですね、現在の販売高は400万ドル前後です。
Well, we are doing around four million in sales right now.

Fig. 6 Annotating a Previous Sentence with Context Related to the Current Zero-Pronoun Sentence

The last feature enables users to download a parallel corpus that is already annotated with zero-pronoun related labels, which can be either a corpus that has been labeled individually by the user or a corpus with aggregated annotations labeled by multiple 0Past users. For the corpus individually labeled by the user, the system checks and warns the user if unlabeled data can still be downloaded if the user intends to. For the corpus labeled by multiple 0Past users, the tool merges annotations from users that have been confirmed and aggregated manually by the administrators and allows the user to download the final aggregated corpus. Fig. 6 shows the two buttons to download the annotate; note that the aggregated annotation button is disabled when available data has not been provided by the administrators. There are several methods to use annotations from multiple users to derive a single ground-truth dataset, such as majority voting and averaging. We aim to address this issue and implement such methods for improving the crowdsourcing feature in the next version of 0Past; for example, by adding an automated aggregation function.

3.3.2 Zero-Pronoun Annotation

After developing the primary features of 0Past, we annotated each sentence from the BSD and TCE corpora with a zero-pronoun label. We performed this annotation process by navigating each sentence in each conversation and deciding whether a sentence written in Japanese contained an omitted pronoun. If it did, the sentence was labeled “Zero-Pronoun” as a zero-pronoun sentence, “Nope” otherwise. Fig. 5 presents the user interface. Furthermore, if it is a zero-pronoun sentence, we look at the previous sentences in the same conversation and select the sentence(s) that provide context for the omitted pronoun. After the annotation processes were completed, we downloaded the annotations to be used as evaluation sets. The downloaded data are in JSON format, similar to the structure shown in Table 10, with two additional key fields: zeroPronoun (Boolean) and contextSentences (Array). The zeroPronoun field contains information on whether a sentence is a zero-pronoun (true) or not (false). The context–sentence field is an array of numbers/integers containing the sentence number of the previous sentence(s) selected as context(s). Table 11 shows an example of zero-pronoun annotation on the downloaded evaluation set based on the BSD corpus development set.

Table 11 Example of zero-pronoun annotation on a chat sentence

An example of an annotated sentence

```
{
  "en_sentence" : "Well, we are doing around four million in sales right now.",
  "en_speaker" : "Mr. Ben Sherman",
  "ja_sentence" : "そうですね、現在の販売高は 400 万ドル前後です。",
  "ja_speaker" : "ベン　シャーマンさん",
  "no" : 16,
  "zeroPronoun" : true,
  "contextSentences": [14]
}
```

3.4 Experiments

This section describes the structure and results of the experiments. In Section 3.4.1, we describe the datasets used in the various stages of our experiments. For annotation and evaluation purposes, this study used two parallel corpora containing spoken conversational sentences. The first evaluation corpus was taken from the development set of the Business Scene Dialogue (BSD) corpus [53], which contains conversations occurring in the business domain. The second corpus is a collection of parallel sentences that we assembled, the Travel Conversation Exercise (TCE) corpus, which contains example conversations that typically occur when traveling to foreign countries.

Previous sections show how we annotated two sets of Japanese-English parallel corpora using 0Past. We created two evaluation sets to assess the MT model when translating sentences that contain anaphoric zero-pronouns, particularly from Japanese to English. In section 3.4.2, using the annotated evaluation set, we trained a model to classify the zero-pronoun label, given a pair of Japanese and English sentences, by fine-tuning the XLM-RoBERTa sequence classification model. This shows the potential for an improvement to 0Past, enabling the tool to provide pre-annotations that can assist annotators, particularly when handling a large number of sentences.

Section 3.4.3 shows how we train our own NMT model for Japanese-English language direction, and evaluate how it perform in comparison with two other publicly available models. In Section 3.4.4, we report our experiments on how we can leverage such zero-pronoun evaluation sets to assess the performance of MT models in a conversational setting. This study uses three NMT models to translate the above evaluation sets, that is, the development set of the BSD and TCE corpora. This experiment aims to compare the performance of existing NMT models when translating conversational sentences with and without zero-pronouns from Japanese to English. The results confirmed that zero-pronoun evaluation sets are essential for supporting the automated evaluation of NMT models.

3.4.1 Data Preparation

For annotation and evaluation purposes, this study used two parallel corpora containing conversational sentences in Japanese and English, which is particularly challenging in terms of the anaphoric zero-pronoun. The first corpus was developed from the Business Scene Dialogue (BSD) corpus. This development set contained 69 business conversations, with 2,051 parallel sentences. The second corpus was a Japanese-English parallel corpus assembled from our laboratory's archive, based on a previous study [66]. The assembled conversations were also used officially to teach English as a foreign language to Japanese university students. We refer to this as the Travel Conversation Exercise (TCE) corpus. It contained various sentences for teaching English as a second language to university students in Japan. There were 888 parallel sentences written in Japanese and English, all of which were sample conversations that may occur when a student travels abroad. Some of the conversation themes included (but not limited to) airports, hotel reservations, sightseeing, and emergencies. Owing to the limited number of sentences, we only used this corpus as an evaluation set and did not include them for training. Both corpora follow a similar format: a JSON array containing many conversations (written in Japanese and English) by two or more speakers. For each conversation, an array contains sentences in the same format.

Additionally, to build the NMT model that used for comparison in Section 4.3, we used the training set of the BSD corpus to train our NMT model; however, as the number of sentences was limited (approximately 20,000 sentences), two additional parallel datasets from other sources were also used to pretrain the model. The first was the Japanese version of the ParaCrawl dataset and JParaCrawl corpus [67]. It is claimed to be the largest publicly available English-Japanese parallel corpus created by NTT Communication Science Laboratories[12]. It contains approximately 10 million parallel sentences and was created by web crawling and automatically aligning parallel sentences. JParaCrawl has been shown to include a wider range of domains than previously available English-Japanese parallel datasets and suitable for training a neural translation model that works well as a pretrained model that can then be fine-tuned to specific domains. The second dataset was the Japanese English Subtitle Corpus (JESC) [68], which contains approximately 2.8 million sentences consisting

[12] http://www.kecl.ntt.co.jp/icl/lirg/jparacrawl/

of sentences from movies and TV subtitles. Unlike JParaCrawl, JESC contains a conversational dialog extracted from movie subtitles and is claimed to be the largest freely available dataset of its kind. JESC[13] was built by crawling subtitles from four free and open subtitle repositories, performing several preprocessing steps to convert them into a form suitable for alignment, and finally combining them into one dataset consisting of 29,368 unique English words and 87,833 unique Japanese words. All three datasets used in this study are publicly available for download from their websites.

3.4.2 Zero Pronoun Sentence Classification for Pre-Annotation

Using the evaluation sets that were previously labeled using 0Past, we conducted an additional experiment to explore the possibility of building a classification model to predict whether a pair of Japanese-English sentences contains an anaphoric zero-pronoun. This experiment aims to determine if we can use the model to provide an automated pre-annotation feature for a new parallel corpus, in which predicted labels can be updated manually by human annotators. The model was trained by fine-tuning the pretrained XLM-RoBERTa [34] base model with a vocabulary size of 250,002 and a maximum length of 512. This experiment used XLM-R primarily because of its capability to handle cross-lingual sentences, public availability, and ease of use. First, we combined the two annotated evaluation sets, the BSD and TCE corpora, to build a collection of 2,939 pairs of parallel sentences, each with its zero-pronoun label. There were 1,762 pairs of sentences marked as zero-pronouns and 1,177 pairs of sentences marked as non-zero-pronouns. The data were then split for training with a train-validation ratio of 80:20, and the model was trained for four epochs with a batch size of 32. AdamW was used as the optimizer, with a learning rate of 2e- 5 and epsilon of 1e-8. SentencePiece was also used by XLM-R to tokenize both Japanese and English sentences, eliminating the need to prepare a specialized tokenizer for each language.

The training phase completed with an average of 32 seconds per epoch, and the trained model performed with 0.88 accuracy on the validation set, meaning that, given a pair of sentences in Japanese and English, the model can correctly predict whether most sentences contain an omitted pronoun. Table 12 shows some of the model's predictions on the sampled pairs of sentences in the test set, and Table 13 shows its predictions on new sentence pairs (those that do not exist in either the BSD or TCE corpora). The combined dataset contained 1,762 zero-pronoun and 1,117 non-zero-pronoun sentences. This experiment, although early, shows another potential benefit by enabling the tool to provide automated pre-annotations on a newly uploaded corpus, reducing the efforts required for manual work but with satisfying results.

Table 12 Zero-pronoun label prediction of sampled sentences from the test set

No.	Sentences		Predicted Label	True Label
1	Japanese	よかった。搭乗手続きをお願いします。	1	1
	English	Good. I'd like to check in.		

[13] https://nlp.stanford.edu/projects/jesc/

2	Japanese	はい、こちらで承ります。	**1**	1
	English	OK. I can handle it for you.		
3	Japanese	航空券をお持ちですか？	**1**	1
	English	May I see your ticket?		
4	Japanese	はい、どうぞ。	**0**	0
	English	Here you are.		

Table 13 Zero-pronoun label prediction of newly created sentences

No.	**Sentences**		**Predicted Label**	**True Label**
1	Japanese	そろそろ雨が降るでしょう。私は私の傘を君に貸しますよ。	**1**	0
	English	It is going to rain. I will lend you my umbrella.		
2	Japanese	明日は学校行きたいんですけど。	**1**	1
	English	But I want to go to school tomorrow.		
3	Japanese	いいえ！	**0**	0
	English	Don't mind it!		
4	Japanese	このりんごを食べていいのかな？	**1**	0
	English	Is it okay to eat this apple?		

3.4.3 Training a Japanese-English NMT for Chat Translation

We conduct experiments to show how to leverage such zero-pronoun evaluation sets to assess the machine translation model performance in a conversational setting. First, we describe the training process for building an NMT to translate conversational sentences from Japanese to English. Our model is based on sequence-to-sequence transformers [3]. Additionally, for comparison purposes and better reproducibility, we used two publicly available pretrained Japanese-English NMT models: the Opus-MT Japanese-English model[14] [4], and the M2M100 model[15] [69]. Unlike our model, these models have been extensively evaluated and trained using a large amount of data and more complex architectures. Finally, this experiment reports the translation performance of the three models for conversational sentences, which were previously annotated using 0Past, with and without zero-pronouns.

All experiments reported in this section were conducted using a PC with a single NVIDIA GeForce RTX 3070 GPU and an Intel(R) Core (TM) i9-10900 CPU with 32 GB of RAM. Owing to limited resources, a considerable gap in performance may exist, measured in the BLEU score, between our own model and state-of-the-art NMT models. This matches our objectives, which do not include building a state-of-the-art translation model, emphasizing the need for phenomenon-specific evaluation for MT and confirming the need for a support tool for building evaluation sets to better support the training and evaluation of future NMT models, focusing on those capable of better anaphoric zero pronoun resolution from Japanese to English.

[14] https://huggingface.co/Helsinki-NLP/opus-mt-ja-en
[15] https://huggingface.co/facebook/m2m100_418M

Our model was based on a transformer model [3]. After experimenting with several hyperparameter combinations [70] [71], we adjusted parameters to fit these limitations. Initially, we trained the models with 16 epochs; however, we gradually reduced the epochs after observing that the model performance did not significantly improve on the test set. The final models were trained with 8 epochs using an encoder/decoder with three and six layers. The model uses an embedding size and feedforward embedding size of 512 with 8 attention heads. For the optimizer, the model was trained using Adam, learning rate was set to 0.0001, and the betas were 0.9 and 0.98, and epsilon was − 1e-9. SentencePiece [21] was tokenized both English and Japanese sentences with a vocabulary size of 32,000.

After training our model, we measured its performance on the BSD and TCE datasets using BLEU scores [72]. The experiments trained several models using three and six encoder-decoder layers and tried several combinations of training datasets, including JParaCrawl only, JParaCrawl+JESC, and JParaCrawl+JESC+BSD corpora training sets along with their backtranslations. Table 14 shows that adding the BSD training set containing approximately 20,000 sentences to the two larger corpora, JParaCrawl and JESC, significantly improved performance. The best performance on the BSD corpus development set was achieved by a combination of three encoder-decoder layers with the JParaCrawl, JESC, and BSD corpus training set and its backtranslations with a BLEU score of 10.2. In summary, we call this model "our final model" in subsequent sections and subsections.

Table 14 NMT model training performance on BSD corpus development set

No. of Encoder and Decoder Layers	**Training Data**	**BLEU Score on BSD Dev. Set**
3 layers	JParaCrawl	7.2
	JParaCrawl ➔ JESC	5.5
	JParaCrawl ➔ JESC ➔ BSD Training Set+back-translation	**10.2**
6 layers	JParaCrawl	5.1
	JParaCrawl ➔ JESC	4.6
	JParaCrawl ➔ JESC ➔ BSD Training Set+back-translation	8.5

We then compared the model's performance with the M2M100 and Opus-MT Japanese to English model to translate the BSD and TCE corpora. Table 15 compares the BLEU scores, which are the averaged results from several runs with different random seeds. Evidently, for both datasets, the BSD and TCE corpora, the Opus-MT model for the Japanese-English translation direction achieved the best BLEU score. This is possibly because, compared to our final model, the Opus-MT model was trained using a larger dataset and architecture; thus, the BLEU score is better, but the inference time is longer. Furthermore, M2M100 is a huge cross-lingual language that can translate various language directions. In contrast, the Opus-MT model we used was trained specifically for Japanese to English, thereby dropping the BLEU score for M2M100 in this direction and increasing the inference time owing to the larger size of the model.

Table 15 Performance comparison between our final model, Opus-MT and M2M100 on BSD development set and TCE corpora

Corpus	**Model**	**BLEU Score**	**Elapsed Time for Inference**
BSD Corpus Development Set (2,051 sentences)	Our final model	10.2	2 m 14 s
	Opus-MT (Ja-En)	**13**	**17 m 3 s**
	M2M100	9.5	40 m 5 s
TCE Corpus (888 sentences)	Our final model	9.7	0 m 42 s
	Opus-MT (Ja-En)	**23.8**	**4 m 34 s**
	M2M100	11.9	11 m 55 s

3.4.4 Results

In addition to measuring the BLEU score and time elapsed during inference, we manually examined some of the translated sentences using the three models. The first sentence in Table 16 is an example sentence without omitted pronouns. All three models can correctly predict the pronoun, as stated clearly in the original sentence; however, the Opus-MT model, which is trained specifically with Japanese-English data and a larger architecture than our model, provided the best result. This matches the BLEU score results from the previous subsection, which confirm that Opus-MT performs better than the other models. However, in the second example, where several pronouns are omitted, our model could translate and predict the translation well, particularly when compared with the M2M100 model. Furthermore, while both sentences provided by Opus-MT and our model translates correctly, the Opus-MT added a pronoun “you” that does not exist in the reference.

Table 16 Sentences with and without zero-pronoun translated using our model, Opus-MT, and M2M100

Non-Zero-Pronoun Sentence	
Japanese (Original)	君はラーメンとお寿司、どっちのほうがすきですか？
English (Reference)	Which one do you like, ramen or sushi?
English (Our Model)	Which sushi are you better?
English (Opus-MT)	Which do you like better, ramen or sushi?
English (M2M100)	Raman and sushi, which of them do you like?
Zero-Pronoun Sentence	
Japanese (Original)	一緒に来れてよかった。見込み客との長い関係を築くと思う。
English (Reference)	**I am** glad to come here together. **I** think **we** will build a long-lasting relationship with potential customers
English (Our Model)	**I am** glad to be here. **I** think **we** will build a long relationship with potential customer.
English (Opus-MT)	**I'm** glad I came with **you**. **I** think **we'll** have a long relationship with potential customers.
English (M2M100)	**I** am very happy to have a long relationship with potential customers.

The results in Table 16 also support the claims made by previous studies [56] [57] [73], in that merely comparing words and characters in translated sentences and their references is insufficient to evaluate MT performance in discourse. An NMT model may perform better in translating an entire sentence to mispredict or even completely miss the omitted pronoun in the original sentence. This also shows that training an NMT model by looking at isolated sentences independently without context could result in poor performance for conversational sentences, even for huge transformer-based models.

As explained in Section 3, this research annotates each sentence in the BSD (development set) and TCE corpora with a zero-pronoun label using 0Past. The BSD corpus contains 1,195 sentences annotated with the zero-pronoun label, whereas the TCE corpus contains 567 sentences annotated with the zero-pronoun label. We separated the translation results of sentences with and without zero-pronouns and calculated the BLEU scores of the three NMT models from the previous sections. Table 17 shows the gap in the BLEU scores achieved by each model when translating sentences with and without the zero-pronoun phenomenon from the BSD corpus. Although M2M100 has the largest gap, 2.4 points in the BLEU score, a similar trend is visible in the two other models, which shows that the score is smaller for zero-pronoun sentences. These results confirm that specialized evaluation sets are necessary to better measure NMT models' performance for language-specific phenomena, particularly anaphoric zero-pronouns in Japanese, which is the focus of this research. The proposed tool is expected to support researchers and practitioners in building evaluation sets with better quality and quantity, with less time and effort.

Table 17 Gaps in BLEU score in translating zero-pronoun sentences in the BSD corpus development set

Model	**BLEU Score**		**Gap**
	Zero-Pronoun	**Non-Zero-Pronoun**	
Our final model	10.05	10.3	0.25
Opus-MT	12.8	13.6	0.8
M2M100	**8.9**	**11.3**	**2.4**

3.4.5 Conclusion

This study proposes a zero-pronoun annotation support tool called 0Past. The primary feature of the current 0Past is the annotation of parallel sentences with zero-pronoun information, given a set of sentences displayed in Japanese and English. To accomplish this, 0Past shows the sentences in a chat-like interface, enabling users to see past utterances inside a conversation to understand the context of the sentence being annotated. Additionally, this experiment created two new zero-pronoun evaluation sets: (1) a Japanese-English parallel corpus called the Travel Conversation Exercise (TCE) corpus, which contains 888 dialog sentences and their zero-pronoun annotations and (2) a zero-pronoun-annotated dataset based on the BSD corpus. This experiment also built a binary classification model with the labeled BSD and TCE corpora to predict whether a sentence contains an omitted pronoun using the XLM-R cross-lingual model. Early results showed that the model achieves an 88% accuracy on the test set and an improvement that may provide automated pre-annotations, which can then be updated by human annotators. Furthermore, this experiment compares three NMT models in translating conversational Japanese sentences into English that contain many omitted pronouns. The results confirm that these evaluation sets are essential to better measure the performance of NMT models for language-specific phenomena, particularly anaphoric zero-pronouns in Japanese. The proposed tool is expected to support researchers and practitioners in building evaluation sets with better quality and quantity, with less time and effort.

Chapter IV A Language Learning Platform for Practicing Text-Based Conversation

Based on the learnings from experiments in the previous sections, this section describes the methodologies and tools that are used to develop the chat-based platform to support second language (L2) acquisition of English, which is focused to support native (L1) Japanese learners. Firstly, this section aims to investigate the challenges faced by learners, mainly for intermediate learners to practice text-based conversations. Secondly, we then propose an interactive web-based software platform, with machine translation, to be used by learners for practicing text conversations in the form of chats and exercises for self-study. The resulting platform enables students to chat with their peers and teachers with Japanese to English translation support and conduct self-study via exercises. It also allows the teachers to guide the students by providing feedback regarding the exercise results. This feedback could then be used to improve NMT model evaluation.

4.1 Related Works

Since mobile technologies have been adopted worldwide, texting has become popular as an essential means of written communication. This is reflected in many aspects of our lives. Companies started adopting platforms, such as Slack, to facilitate more accessible communication between their employees, among other things. Texting is also a popular means of communication among school-aged and college students worldwide, many of whom come from different first language (L1) backgrounds [74] [75], automatically, this includes English language learners (ELLs). Accordingly, many researchers and educators have tried integrating texting into language instruction and self-regulated learning interventions to help second language (L2) students learn different aspects of language [76] [77]. On the other hand, it is argued [78] that translation could be an essential tool for language learning. As with the rise of texting as a way for people to communicate globally across countries and different languages, so does the use of various chat translation skills and tools. Incorporating translation could encourage the learner to search (flexibility) for the most appropriate words (accuracy) to convey what is meant (clarity), which are essential in foreign language learning.

Despite this, many linguists and teachers perceive translation in foreign languages differently, making it a largely criticized and debated topic. One of the main reasons for this is that throughout the years, many studies can be found that have either completely agree or completely ignore the use of translation for teaching a foreign language. In recent years, however, there has been an increasing interest in revisiting the potential of using translation for foreign language learning. It is suggested that, when used properly, translation is a good tool in the English language learning course aimed at enhancement of students' foreign language skills [79]. Learning gets meaningful via translation, and better comprehension promotes foreign language proficiency [80]. The ability to translate gives foreign language learners a sense of accomplishment, so first, learners translate mentally to help them clarify ambiguous or unknown vocabulary meanings. If they fail to do so, they turn to online translating [81], which could provide guidance for active communication or understanding texts. However, current translation tools, such as Google Translate, are often detached from language learning or communication platforms. Users would need to go back and forth between different apps, causing the whole experience less seamless and integrated.

4.2 Overview and System Architecture

Our proposed system is a web-based platform that can be accessed using a range of devices, such as laptops and smartphones, as long as it has an internet connection. We call it Chappin, which stands for Chat Platform for Practicing English.

4.2.1 System Architecture

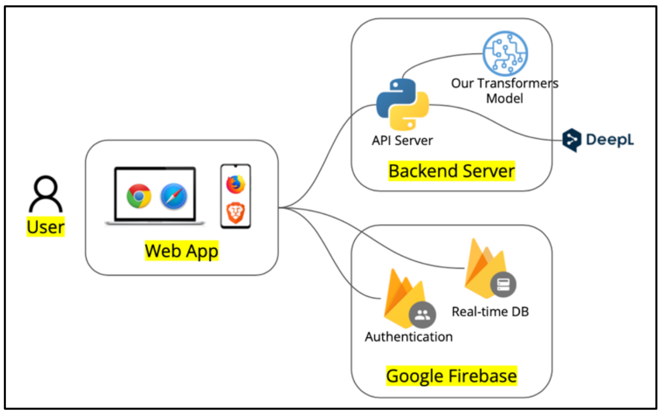


Fig. 7 Architecture of the proposed system

We use Quasar[16] to build the frontend of our proposed system. It is based on Vue.js 6 (version 3) with the Composition API. Quasar components are responsive by default, making it easier for us to provide user interfaces for various device types and sizes. Most devices with an internet connection and a browser application could access the application with a responsive layout and interfaces tailored to various screen sizes. In the future, we could extend our application to be compiled as a mobile application, running on Android and iOS devices, using commands provided by the Quasar CLI. Our system uses the cloud services provided by Google Firebase7 for authentication and database. All the data the application uses is stored using the Firebase Real-Time Database, and email authentication is implemented using the Firebase Authentication. We use Python (version 3.9) to build the backend server, performing various pre-processing steps such as input normalization and tokenization, calling an external translation service (DeepL API[17]), calculating scores, and serving our Transformers-based

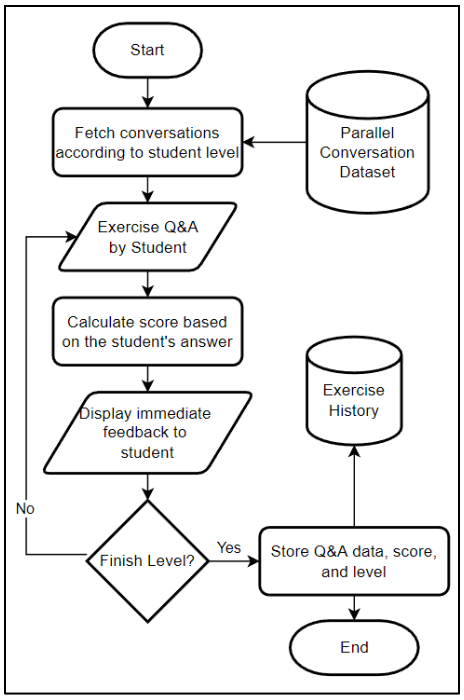
[16] https://quasar.dev

[17] https://www.deepl.com/docs-api

neural translation model (Ja→En). Fig. 7 shows the overall diagram of the system architecture of our proposed system.

4.2.2 Application Feature and Design Flow

In this paper, we report two main features of our proposed system: a chat feature with in-app auto-translation from Japanese to English and an exercise feature for self-practice and assessment. This section explains the design and flow of these features. First is the chat feature with in-app auto-translation from Japanese to English. The motivation is to encourage the learners to text in English, but often it is hard to produce words. This is particularly interesting for intermediate learners who can understand while reading English but are stuck when writing English. We implement the DeepL translation tool for this, automatically translating any Japanese sentences to be sent into English. After the translated messages are sent, the users in the chatroom can still see the original message by clicking the "see original" button. Some popular services worldwide, such as Grab[18], have used this feature to overcome the language and communication barrier between users of different languages. One typical case in a Japanese university is an activity involving both international and Japanese students. These activities could be an excellent chance for intermediate to advanced English learners to practice their English. However, not many are confident in their skill, even with text messages, and going back and forth between chat platforms and translation apps to send a chat message could be time-consuming. Providing a chat platform that provides in-app automatic translation could encourage them to be more active in using English.


Start
Fetch conversations according to student level
Parallel Conversation Dataset
Exercise Q&A by Student
Calculate score based on the student's answer
Exercise History
Display immediate feedback to student
No
Finish Level?
Yes
Store Q&A data, score, and level
End

[18] https://www.grab.com/

Fig. 8 Flowchart to score users answer

The first feature enables the users to communicate with other users of the same system, which can be their learning peers, teachers, or any external party registered to the system. On the other hand, the second feature which contains self-practice and assessment enables the users to conduct conversation practice by themselves. Similar features can be found in other learning applications such as Duolingo; however, in our case, we focus mainly on conversation practice by providing a conversational scenario that could happen in real life in the form of text messages. Fig. 8 describes the flow of this conversation exercise feature. When the user starts the exercise feature, the app will fetch conversations from the database depending on the user level, then display them individually. After the user submits their answer corresponding to a particular question, the application will calculate the score and provide text feedback. Finally, after the user finishes a level (consisting of 5 conversations/questions), the answers, score, and level data are stored in the database.

4.3 Implementation

As described in the previous section, our system's two main user-facing features are the chat feature with in-app auto-translation and the exercise feature for self-practice and assessment.

4.3.1 User Chat Feature with In-App Translation

Chatroom feature with in-app auto-translation aims to encourage active use of English in a conversation and enables the users to have real communication with others. Students can also create a chatroom to talk with the teacher. A sample use case for a language learning class in a Japanese university is an activity involving both English-speaking international students and Japanese students. These kinds of activity could be an excellent chance for intermediate to advanced English learners to practice their English. However, simply asking the students to communicate might not go smoothly as many learners are not confident in their skill. Even with existing translation tools, using a separate application/platform causes the student to go back and forth between chat platforms and translation apps which could be inconvenient.

Fig. 9 shows the chat feature's user interface with in-app auto-translation viewed on a mobile browser. As seen on the left screen, all chat messages are written in English. However, there is a "See Original Text" button in some chats. Chat balloons containing this button mean that the message is initially sent in Japanese; thus, if the readers (or sender) want to view the original message written in Japanese, they can. The original text is shown by replacing the English text, as shown on the right screen of Fig. 9.

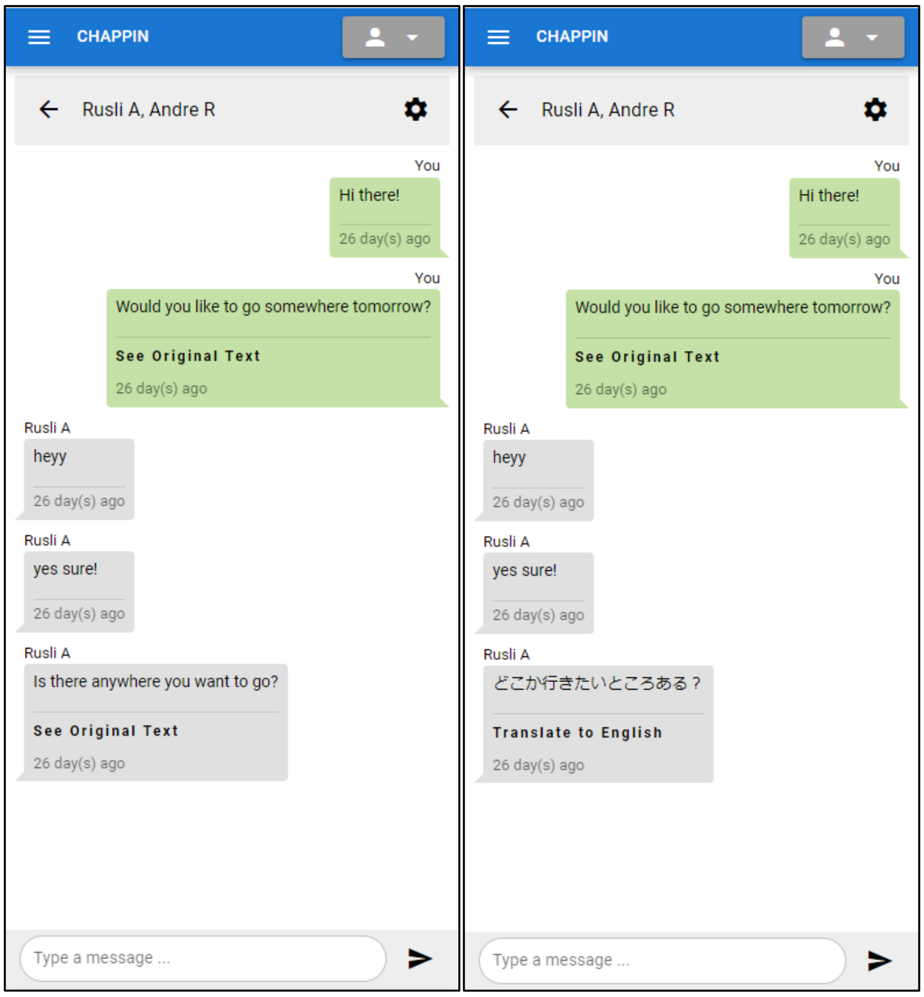
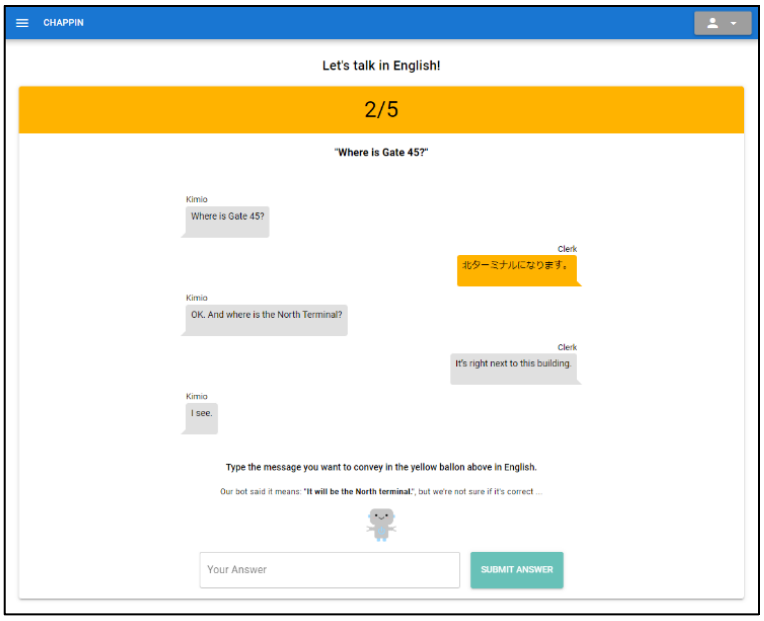


Fig. 9 User interface of the chat feature with in-app translation

4.3.2 Self-Exercise with Automated Question-Answer Generation and Feedback

Next is the exercise feature aimed at self-practice of understanding and engaging in conversational English. This exercise feature is divided into two types of quizzes, both aiming to help students practice understanding text dialogs between several parties. The first type of exercise is a dialog translation exercise, and the second is a C-Test exercise. Both exercises in this feature automatically generate question-answer pairs given a parallel dataset of conversations and then use an automated scoring system for immediate feedback for the students, which can then be checked and corrected manually by the teachers for higher quality feedback.

Fig. 10 depicts the user interface of the dialog translation exercise where a series of chats in English are displayed to the users, with one chat balloon written in Japanese. This exercise asks the users to translate the yellow chat balloon into English. This exercise is not just a simple sentence translation task because the users need to understand the conversation to get the context and write the correct translation. For example, in Japanese, there is the anaphoric zero-pronoun sentence in which the pronoun is not explicitly said in a sentence, however, when translating to English, users need to insert the pronoun.


CHAPPIN

Let's talk in English!

2/5

"Where is Gate 45?"

Kimio
Where is Gate 45?

Clerk
北ターミナルになります。

Kimio
OK. And where is the North Terminal?

Clerk
It's right next to this building.

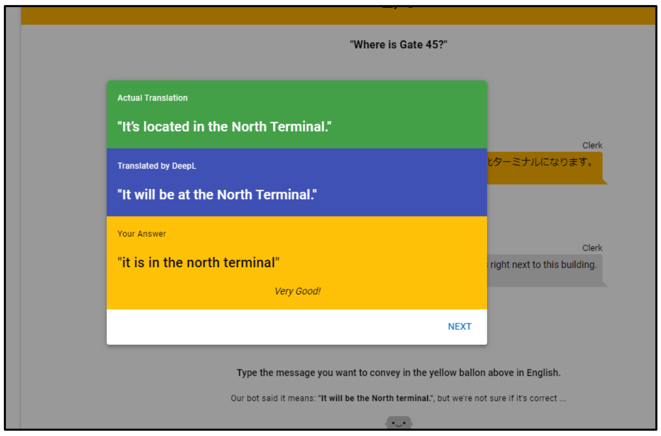
Kimio
I see.

Type the message you want to convey in the yellow ballon above in English.

Our bot said it means: "**It will be the North terminal.**", but we're not sure if it's correct ...

Fig. 10 User interface of the translation exercise feature

Choosing the correct pronoun requires the users to understand the context. In addition, there can be a guide translation above the answer input field, which uses an NMT model for translating sentences from English to Japanese, however, with limited performance. We trained the NMT model used for this purpose in the previous section. It is based on the sequence-to-sequence Transformers architecture with three layers of encoders-decoders and 8 attention heads. Using this guide translation, users can get a hint of what kind of English sentences they need to write but still think for themselves, as the translations are often not 100% accurate, and they cannot understand contexts. Within one level, there are five questions, each containing a particular conversation from a dataset.


"Where is Gate 45?"

Actual Translation
"It's located in the North Terminal."

Translated by DeepL
"It will be at the North Terminal."

Your Answer
"it is in the north terminal"

Very Good!

Clerk
北ターミナルになります。

Clerk
right next to this building.

Type the message you want to convey in the yellow ballon above in English.

Our bot said it means: "**It will be the North terminal.**", but we're not sure if it's correct ...

Fig. 11 Score feedback of the translation exercise answer

After the users write their English sentences corresponding to the conversation in the yellow balloons, the user can press the submit button. Fig. 11 shows the immediate feedback provided by the application to the users for the translation exercise answer, we use reference sentences from two sources. The first is the original English reference from the datasets used in the application, and the second is the sentence translation result using the DeepL translation API. In our experiment, we use two parallel datasets as the source of sentences. The first dataset is the Travel Conversation Exercise dataset, and the second is the Business Scene Dialog (BSD) Corpus [53]. Initially, the answer submitted by the user is then compared with the two references, and a score is calculated using the BLEU score. The text feedback will be displayed depending on the calculated BLEU score, for example, feedback that says, "Very Good!". However, after we conducted user interviews and a further experiment on NMT evaluation metrics (described in Chapter V), we revisited Chappin and changed the scoring system to use COMET [82].

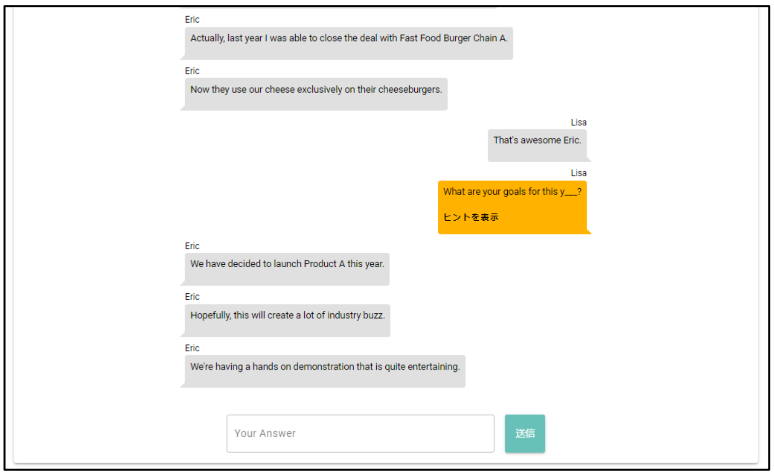

Eric
Actually, last year I was able to close the deal with Fast Food Burger Chain A.
Eric
Now they use our cheese exclusively on their cheeseburgers.
Lisa
That's awesome Eric.
Lisa
What are your goals for this y___?
ヒントを表示
Eric
We have decided to launch Product A this year.
Eric
Hopefully, this will create a lot of industry buzz.
Eric
We're having a hands on demonstration that is quite entertaining.

Fig. 12 User interface of the C-Test exercise on a desktop browser

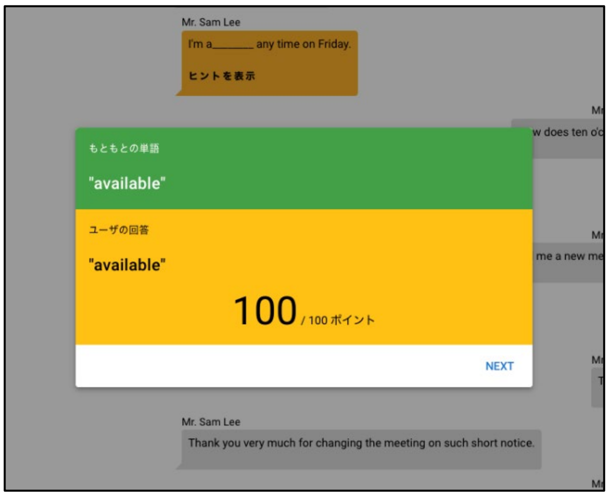

Mr. Sam Lee
I'm a_______ any time on Friday.
ヒントを表示
もともとの単語
"available"
ユーザの回答
"available"
100 / 100 ポイント
Mr. Sam Lee
Thank you very much for changing the meeting on such short notice.

Fig. 13 Screen showing the score of a submitted answer on the C-Test exercise

Fig. 12 shows the C-Test exercise, in which, similarly, a series of chats in English is displayed, but instead of translating, the users are asked to fill a blank part with a word/phrase. Similarly with the translation exercise, the system will randomly choose one chat message from a conversation. However, here, we can set additional rules regarding the length of the selected sentence. By default, we set that the randomly selected chat message must have more than 4 words and there should be at least one word with 4 letters which will be the question. These default numbers are selected arbitrarily based on a preliminary user test with a few participants, but the proposed system allows teachers to adjust this number accordingly. The selected chat message will then be displayed in different color, and the selected word will be hidden (the question), as can be seen in Figure 12. Here, we use a simple string matching to check whether the word entered by the user matches the actual hidden word and then display the score as shown in Fig. 13.

4.4 Usability Tests

While we are developing the language learning platform explained previously, we conducted two separate user tests. The first one is conducted as a preliminary user test when the platform was not 100% complete, and the second one is conducted after the platform development is completed.

4.4.1 User Test: First Phase

During this preliminary user test, Chappin only has the Chat Feature with In-App Translation and the Translation Exercise without automated question generation. For the immediate feedback for scoring students answer, it uses BLEU score. We deployed the application on a private server and asked two participants to conduct early user testing and get feedback. Both participants are graduate students with L1 Japanese who can read and write formal English (e.g. research papers) but are not used to using English for two-way conversations. The participants are willing to practice conversational English to prepare themselves as they will use it a lot when they start working in the industry after graduation. After using our proposed system for some time, we asked for feedback from these participants to measure the acceptance of our system, mainly focusing on the perceived ease of use and perceived usefulness aspects, which are well-known aspects for technology acceptance tests. Along with interviews, we asked the participants to rate both aspects on a scale of one to five, with one being very bad and five very good. On the perceived ease of use aspect, both participants rated 5, meaning that the application is perceived as easy to use. On the other hand, the participants rated 3 and 4 for the perceived usefulness of the application. Overall, the application is perceived to be useful for English learners to practice text-based conversations. However, based on the feedback and discussion with the participants, some concerns need further analysis.

There are two main concerns in this chat feature with in-app auto translation. The first one is the challenge for the machine to understand context, especially in chats. One sentence written in the source language can often be translated into different versions of sentences in the target language. This is especially important in text messages, where contexts are essential, and the contexts often exist in the previous chats. For example, the sentence 「来る前に食べたの？」 which means "did [pronoun] eat already?", does not have a pronoun. In this sentence, the translation can be "did you eat before coming?" or "did she eat before coming?" depending on the context. The second one is the risk of overusing the translation service if the students do not try to practice their English. However, while we need to conduct further experiments to improve the model to better understand contexts and observe how the users utilize the auto-translation feature, we argue that the current imperfection of translation models in understanding context can be a catalyst to encourage the users to try writing English by themselves. Since the translations provided by the machine are often not perfect and can lead to misinterpretation by the readers, the more advanced users might as well try their best to write the messages by themselves.

4.4.2 User Test: Second Phase

In this next user test, the platform is finished and improved, also considering the feedback from the preliminary user test. During this test, the platform has the Chat Feature with In-App Translation and the Exercise Feature with two types of quizzes, the translation exercise and C-Test exercise. Firstly, we conducted a direct observation with three university students in Japan who are intermediate English learners. The setup is as follows:

1. The direct observation experiments with each participant are conducted on a separate time, one by one.
2. Firstly, the participants are given a preliminary questionnaire, asking about how they feel about text-based conversation in English.
3. Next, we introduce them to the proposed system by explaining each feature without demonstrating the system because we want to know whether the user interfaces are easy to understand for the users or not.
4. After the participants understand how the features should work and how they can start using the system, they are asked to freely use the system to exercise for a minimum of 10 minutes. During this time, we closely observe what the users are doing with our system and the users are asked to think aloud. The users are also allowed to ask questions if they are stuck with something.
5. After the observation, the participants are asked to voice their opinion via an interview session which lasts around 10 minutes in average.
6. Lastly, we asked the participants for their comments on the question and answers generated with ChatGPT and how the scoring of the translation exercise can be improved with COMET for the next version of our proposed system.

In parallel, we also distributed Chappin to an English class consisting of 25 Japanese university students and gathered feedback using online forms.

Our user test experiment with the proposed system indicates several key findings. Firstly, based on the preliminary survey, the students expressed a desire to practice text-based conversation but reported a lack of opportunities to do so. Being in an environment with only a few English speakers makes it hard to engage with English conversations and chatting to strangers in online forums or chat platforms seems too daunting. From the observation of the users, the system was found to be easy to use, with users having no difficulties in navigating between pages and features within the system. Additionally, the convenience of being able to access the system through both personal computers and mobile devices was noted. On one hand, users preferred using PC screens due to the wider and clearer display, while on the other hand, the mobility of using smartphones was deemed useful for accessing the system outside of home.

The self-assessment features platform, which provides immediate feedback on both exercises, are seen as an important part of the system. This is an advantage compared to traditional language classes, where students usually need to wait days or even weeks for teacher assessment to complete and check their own performance. However, it is crucial for the system to provide reliable scores, which can be challenging in language learning. For example, when scoring translations, there are times where the students translated correctly but using different words than the actual English sentence, leading to a low score given by the system. This might be not problematic for advanced English speaker because they could still be confident with their answers and understand that it is a limitation of the system. However, lower-level students who may not understand may feel discouraged despite their efforts. Lastly, based on the interviews, even though the proposed system is very useful, and they are looking forward to more of these tools integrated in classes, the role of teachers in language learning is still very important, especially for lower-level students. The participants explained that they can enjoy using this tool because they have reached a certain level in English, even so, ideally it would be best if there is still a teacher guiding them with a suitable program and curriculum, instead of blindly practicing by themselves. Additionally, a potential challenge was also identified with regards to student motivation. While students were initially excited about conducting both types of exercises in the proposed system, few of them found it monotonous after some time. The various questions generated by AI can be considered interesting and could potentially keep students motivated, we explore this possibility in Chapter 5, using ChatGPT.

4.5 Conclusion

In this study, we proposed and developed a computer-assisted language learning platform, namely Chappin, to support Japanese intermediate English learners to practice text-based conversation which is a highly relevant skill in the global industry these days. The proposed system enables the user to conduct self-practices with automatically generated questions and get immediate feedback to conduct self-assessment, especially when they are outside classes and teacher’s support is limited. Based on the experiments, the proposed system is easy to use and considered to be useful for intermediate students to practice English.

Chapter V Additional Experiments

This chapter describes some additional experiments that we conducted in addition to the main experiments explained in previous chapters.

5.1 Data Collection for Improving NMT Evaluation

As techniques for building neural machine translation systems improve, evaluating them has become more critical than ever. Currently, there is a lack of consensus and standardization concerning translation quality assessment— both human and machine—given the complicated cognitive, linguistic, social, cultural, and technical process this supposes [83]. Human evaluation techniques are still perceived to be more dependable than automated metrics. A recent technique for human evaluation is MQM [84], an elaborate error-based methodology for scoring output, typically carried out by skilled human annotators. Although the results of human metrics are often considered more reliable than those provided by automated metrics, it requires significant demands on time and resources and is hard to reproduce [85]. On the other hand, evaluation using automated metrics require less human effort; thus, more objective and less costly than human evaluation.

One of the most popular metrics for evaluating machine translation performance is the Bilingual Evaluation Understudy (BLEU) [72], a precision measurement carried out at n-grams, indivisible language units. It employs a modified precision that considers the maximum number of each n-gram appearance in the reference translation and adds a brevity penalty to the measurement calculation. Many studies have also experimented with improving the original BLEU metrics, such as by implementing various smoothing techniques for sentence- level BLEU [86]. However, most automated metrics require reference translations. Since these metrics evaluate translation in relation to its similarity to the reference translations, we need more of them to get more reliable measurements of the actual quality of the translated sentences. In many cases, not enough of them are available, even though it is acknowledged that there is no single correct translation.

5.1.1 Experimental Setup

When Chappin, the proposed platform in described in the previous section, is used in a learning environment where a teacher (or teachers) exists, the teacher can check the students' answers and manually assess the answers. This is considered essential in a learning platform since even though automated feedback can be quick and interactive, there are times when the assessment is not accurate. The teacher's role is crucial in ensuring that the assessment is done correctly and providing guidance to the learners. Furthermore, the system will get the verified translations and update the parallel dataset in the database for translations that the teachers have already verified. Fig. 14 describes the flow of this process. After updating the dataset, for one Japanese sentence, there can be more than one English translation. This can potentially improve the dataset to be used as an evaluation set of NMT models, especially for metrics like the BLEU score, which requires reference translations to be compared with the predicted translation.

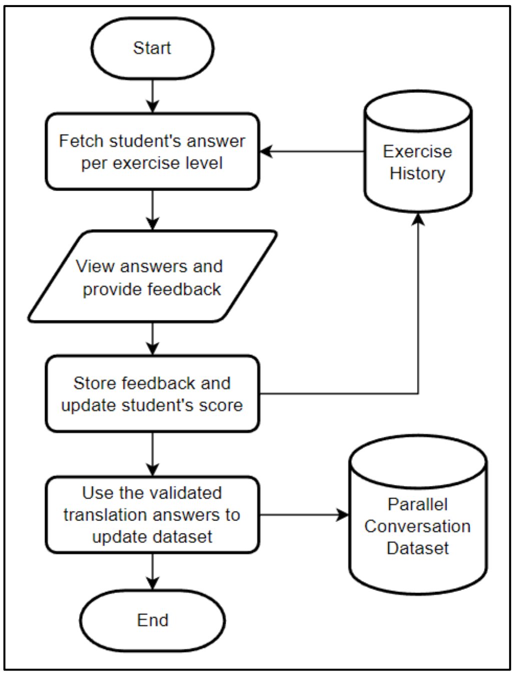

Start
Fetch student's answer per exercise level
Exercise History
View answers and provide feedback
Store feedback and update student's score
Use the validated translation answers to update dataset
Parallel Conversation Dataset
End

Fig. 14 Flowchart of updating the reference dataset with newly verified translation

5.1.2 Results and Discussion

We use some of the verified translation answers from the participants in this early user test and add them as a reference translation to the dataset. Table 18 shows an example with different BLEU score results when using the original parallel sentences and the updated sentences. Here, we use the sentence_bleu() function from nltk with a smoothing method 7, which is tailored for sentence-level BLEU, to calculate the score. The source and target sentence in the original parallel sentence is taken from the BSD corpus, and the answer is provided by one of the two early users in our experiment. The answer says, "There are more than 100 here". However, using the original dataset, it only scores 20.423. This is not surprising because BLEU compares the word n-grams with the existing target sentence (references). In the updated parallel sentence, we added another target sentence that can be used as a reference for a correct translation, and the BLEU scores improved to 88.286.

Table 18 Different BLEU score with newly added reference translations

Original Parallel Sentence	
Source Sentence (Ja)	**BLEU** 20.423
"これで 100 以上あるよ"	
Target Sentence (En)	
"There are over a hundred here."	

Answer (En)	
“There are more than 100 here.”	
Updated Parallel Sentences	
Source Sentence (Ja)	
“これで 100 以上あるよ”	
Target Sentences (En)	
“There are over a hundred here.”	**BLEU** 88.286
“With this, there are more than 100.” ← *new*	
Answer (En)	
“There are more than 100 here.”	

5.2 Comparison of Various NMT Evaluation Metrics

In MT, many traditional baseline metrics remain popular for evaluating MT systems due to their lightweight and fast computation [82]. However, as MT systems improve over time, commonly used metrics struggle to correlate with human judgment at the segment level and fail to evaluate the highest-performing MT systems adequately, thus misleading system development with incorrect conclusions. Among these metrics, BLEU is often considered the de facto standard of MT evaluation metrics. In advanced translation, however, there are many cases in which the description of phrases differs but implies the same meaning. In these cases, BLEU will struggle to perform adequately, even if there is a system that can translate with high quality. It is noted that even though BLEU is fast and can be helpful to assist researchers and developers in quickly performing initial experiments, BLEU should not be the primary evaluation technique in NLP papers [87]. Furthermore, it is argued that, due to its limitations, the common use of BLEU over the past years has negatively affected research decisions in MT [88].

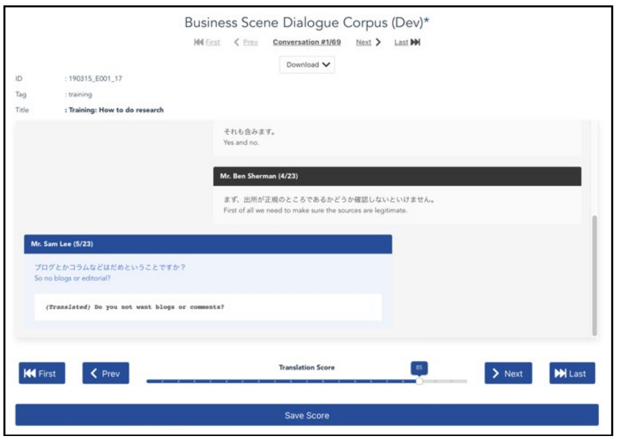
In recent years, many efforts have been conducted to propose better metrics that can measure the quality of MT systems, such as BERTScore [89] and COMET [82], which are proven to correlate better with human judgments. However, only a few provided analysis on their use for evaluating chat translation, which is notably challenging for the Japanese language with its various challenges, such as the anaphoric pronoun resolution, that occurs more frequently in spoken language than in written language [53]. Furthermore, many works that try to build models for this are evaluated using BLEU, which is often argued as insufficient for advanced models [90]. However, the performance of many recent neural-based metrics has yet to be analyzed. This paper compares and analyses results from different models measured with various metrics and how these metrics correlate with human judgment.

This experiment compares how different evaluation metrics perform on chat translation from Japanese to English. Then we analyze and emphasize the importance of using suitable metrics for measuring the performance of our translation systems depending. Additionally, we provide datasets containing the translation results of the Japanese-English Business Scene Dialogue corpus by three NMT models and a set of human-annotated scores of one of the model's translation results.

5.2.1 Experimental Setup

We use three NMT models to translate conversations from Japanese to English and store the translation results. Then, we calculate the score of each model's translation with various evaluation metrics, from traditional and commonly used metrics like BLEU to more recent neural-based metrics such as BERTScore. Furthermore, we compile human-annotated scores of machine-translated sentences from one model, measure how each metric correlates with human scores, and share our findings.

To get the human-annotated score, we built a direct assessment tool for the manual evaluation of machine translation. The tool enables the user to view the chat dataset in a familiar chat-like user interface displaying the current segment/chat conversation along with its document-level context (previous chats), then score each translation from 0 (wrong) to 100 (perfect). It is developed following the direct assessment tool used for human evaluation on the WMT 2020 Shared Task on Chat Translation [62]. Fig. 15 shows the user interface of this tool.



ID : 190315_E001_17
Tag : training
Title : Training: How to do research

それも含みます。
Yes and no.

Mr. Ben Sherman (4/23)

まず、出所が正規のところであるかどうか確認しないといけません。
First of all we need to make sure the sources are legitimate.

Mr. Sam Lee (5/23)

ブログとかコラムなどはだめということですか？
So no blogs or editorial?

(Translated) Do you not want blogs or comments?

Fig. 15 User interface of the direct assessment tool

5.2.2 Results and Discussion

In the first experiment step, we use the Business Scene Dialogue (BSD) corpus [53]. We translated the whole development set of the BSD corpus (2,051 chats) using three models, namely our own Transformers-based MT model, M2M100, and MarianMT for Japanese to English language direction. Previously, only BLEU was used to evaluate the results; in this paper, we use six metrics containing both traditional baseline metrics, which are BLEU [72], TER [91], METEOR [92], and chrF [93], and recent neural-based metrics (BERTScore and COMET). We can see in Table 19 that even though each metric uses different approaches in evaluating the translated results, all metric seems consistent when deciding which performs the best of the three models. In other words, traditional metrics are still valid if we only want to rank which models perform better than the others, and the processing time is essential. Table 19 shows that all metrics indicate MarianMT outperforms the other two models.

Table 19 NMT models performance measured by different metrics

Metric	**Ours**	**M2M100**	**MarianMT**
BLEU	0.14	0.12	**0.17**
TER	79.70	78.62	**75.54**
METEOR	0.41	0.39	**0.47**
chrF	34.31	35.07	**39.77**
BERTScore	0.92	0.92	**0.93**
COMET	-0.022	0.026	**0.225**

Using the direct assessment tool for manual evaluation, we collected human-annotated scores of 10 conversations containing 283 chats inside the development set of the BSD corpus, translated by our own MT model. We asked two human annotators to score the translation and averaged the results. Both annotators speak Japanese and English, one is a native English speaker, and the other is a native Japanese speaker. The conversations are picked arbitrarily from the development set. Table 20 shows how each metric correlates with the human-annotated score for each chat translation. Neural-based metrics pre-trained with language models outperform traditional baseline metrics in correlation with human scores. It is worth noting that neural-based metrics slow down when the computation is done using the CPU only. This can be a limitation when we still experiment with architectures and hyperparameters in the initial phases of training an MT model. By calculating the Pearson correlation coefficient (r), it can be seen that more recent metrics correlate better with human-annotated scores than traditional baseline metrics, with COMET achieving the highest correlation.

Table 20 Pearson's r on each metric

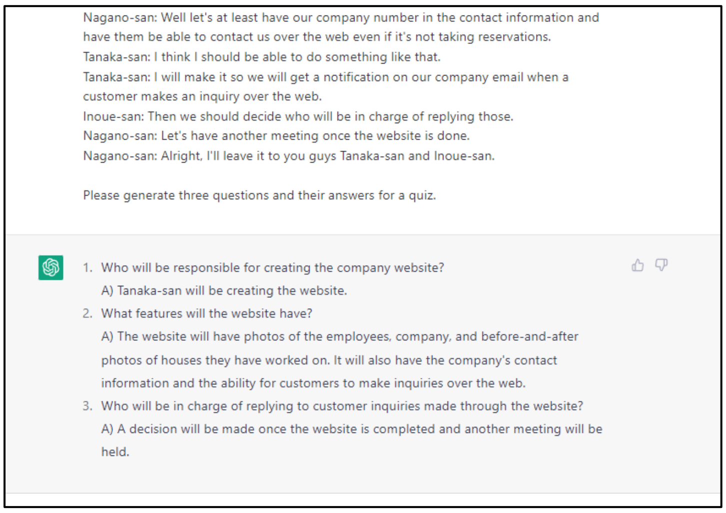
Metric	**Pearson's r**	**Elapsed Time**
SentenceBLEU	0.2246	**1.06 s**
TER	-0.1995	1.19 s
METEOR	0.2860	1.23 s
chrF	0.3038	1.07 s
BERTScore (F1)	0.4342	GPU: 6.54 s CPU: 98.73 s
COMET	**0.5246**	GPU: 16.71 s

		CPU: 220.71 s

5.3 ChatGPT for Automated Question-Answer Generation and Scoring

5.3.1 Experimental Setup

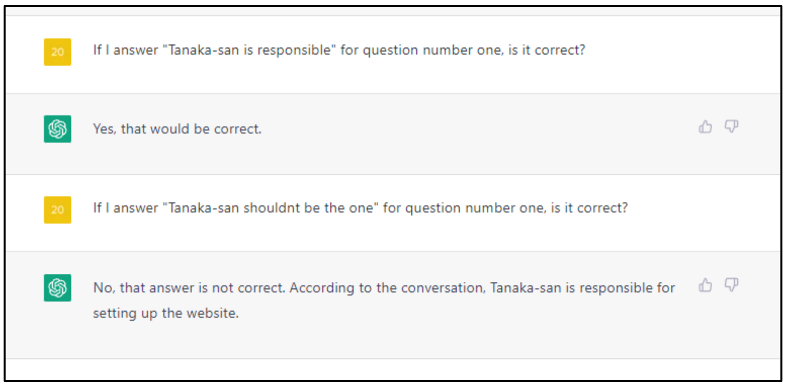
In the previous chapter, it is shown that, given a conversational dataset, we can generate questions using rule-based methods in natural language processing. However, it can also be seen that the types of questions are close-ended and limited in forms. We explored the use of ChatGPT [94] to improve the quality and creativity of the generated questions. The conversations in our proposed system are restructured and prepared to be used as a prompt for ChatGPT. For example, we prepare each chat messages in the format of "Speaker Name: Chat Message" separate each chat message in a conversation, add "Based on the conversations below:" above the conversation, and add "Please generate three questions for a quiz" below the conversation. Fig. 16 shows a sample response from ChatGPT. Like the example below, we tried prompting ChatGPT with many conversations from different datasets and it can accurately create relevant questions to the given conversation, which shows that it has the potential to enhance the current platform for language learning. Furthermore, we can also prompt it to check whether a user answer to a particular question it previously generated is correct or not, as can be seen in Fig. 17.


Nagano-san: Well let's at least have our company number in the contact information and have them be able to contact us over the web even if it's not taking reservations.
Tanaka-san: I think I should be able to do something like that.
Tanaka-san: I will make it so we will get a notification on our company email when a customer makes an inquiry over the web.
Inoue-san: Then we should decide who will be in charge of replying those.
Nagano-san: Let's have another meeting once the website is done.
Nagano-san: Alright, I'll leave it to you guys Tanaka-san and Inoue-san.

Please generate three questions and their answers for a quiz.

1. Who will be responsible for creating the company website?
   A) Tanaka-san will be creating the website.
2. What features will the website have?
   A) The website will have photos of the employees, company, and before-and-after photos of houses they have worked on. It will also have the company's contact information and the ability for customers to make inquiries over the web.
3. Who will be in charge of replying to customer inquiries made through the website?
   A) A decision will be made once the website is completed and another meeting will be held.

Fig. 16 A sample response by ChatGPT when prompted to generate questions based on a conversation


20

If I answer "Tanaka-san is responsible" for question number one, is it correct?

Yes, that would be correct.

20

If I answer "Tanaka-san shouldnt be the one" for question number one, is it correct?

No, that answer is not correct. According to the conversation, Tanaka-san is responsible for setting up the website.

Fig. 17 Sample responses by ChatGPT when prompted to check whether an answer is correct or not

5.3.2 Results and Discussion

The various questions generated by ChatGPT are considered interesting and could potentially keep students motivated. We can see that language models have emerged as a promising tool for language learning, particularly for those learning English as a foreign language. These models can generate human-like text based on a given prompt or conversation, which can be beneficial for students in their language practice and comprehension. Moreover, these models can be used to create educational content and personalize learning experiences, making them highly relevant for language learners. However, despite the potential benefits of large language models, the role of teachers remains crucial in language learning. Teachers are responsible for guiding and monitoring student learning, as well as ensuring that students are learning in a meaningful and effective way. In language learning, teachers play a key role in helping students develop the necessary language skills and competencies, as well as fostering critical thinking and cultural understanding.

Additionally, large language models may have limitations and unexpected brittleness, and it is the teacher's responsibility to ensure that students are aware of these limitations and have the necessary competencies and literacies to understand the technology and its implications. Teachers should also provide students with clear strategies for fact-checking, as well as encourage critical thinking and independent learning. Lastly, large language models should be integrated into educational systems and teaching curricula in a way that aligns with a clear pedagogical approach. This means that teachers should be involved in the design and implementation of large language models and should be provided with the necessary training and support to effectively integrate these models into their teaching practices.

Chapter VI Conclusion

In recent years, the NLP and machine learning fields advanced rapidly that new methods and models are released frequently. It is critical for researchers and practitioners in the educational field to be able to recognize which methods and techniques could be used to in their respective field. This study experimented with various recent and popular natural language processing (NLP) techniques to support second language acquisition of English, for L1 Japanese students. We started off from various techniques to handle non-English text data, focusing on Japanese morphological analysis methodologies. We compared three popular tokenization tools, MeCab, Sudachi, and SentencePiece that all can be used to handle Japanese text and experimented with them to get an understanding of how we can leverage them in building a language learning system. Then, we surveyed existing cross-lingual model, and then experimented with XLM-RoBERTa for handling cross-lingual text with transfer learning, and then compared its performance with other models from related works of research.

From there, we moved on to using sequence-to-sequence Transformers and SentencePiece to build our own neural machine translation (NMT) model. We tried building using relatively minimum resources in terms of computational cost and dataset, and produced a model which is lightweight and fast in translating Japanese to English conversational texts, in comparison to several existing model. Despite the quick translation ability, the translation quality could not keep up with the other model which is trained with more data and deeper architecture. Finally, we proposed and developed a computer-assisted language learning platform to support Japanese intermediate English learners to practice text-based conversation which is a highly relevant skill in the global industry these days. The proposed system enables the user to conduct self-practices with automatically generated questions and get immediate feedback to conduct self-assessment, especially when they are outside classes and teacher’s support is limited. Based on the experiments, the proposed system is easy to use and considered to be useful for intermediate students to practice English.

The use of computer-assisted language learning (CALL) and large language models (LLMs) has the potential to revolutionize the way English as a foreign language is learned and taught. Researchers and developers should always put the human first when developing these systems, as no matter how advanced the technologies behind the system, if it is not used accordingly by the users due to misunderstanding or inability to do so, it will be less meaningful. Furthermore, the role of the teacher in language learning remains crucial. Additionally, a clear strategy within educational systems and a strong pedagogical approach that focuses on critical thinking and fact checking are necessary to fully integrate and take advantage of large language models in learning settings. Finally, while these models can provide access to vast amounts of language data, they can also generate errors or unexpected results that can mislead students. As such, it is essential for teachers to monitor and guide students' use of these models to ensure that they are using the models in an effective and meaningful way. Lastly, the goal of these kind of research which integrated AI in education should not be to eliminate the role of teachers, but to support them. Together with various advances in AI, we can advance the language learning field and provide better education for the students.

References

[1] R. Selke, T. Sekiguchi, A. Moehle, A. Elsharqawy and P. Streich, "Foreign Language Proficiency as an Asset for Japanese Graduates," *IAFOR Journal of Education,* vol. 6, no. 1, pp. 103-120, 2018.

[2] I. S. P. Nation, Learning Vocabulary in Another Language, 2 ed., Cambridge: Cambridge University Press, 2013.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention Is All You Need," in *arXiv preprint arXiv:1706.03762.*, 2017.

[4] J. Tiedemann and S. Thottingal, "OPUS-MT – Building open translation services for the World," in *The 22nd Annual Conference of the European Association for Machine Translation*, Lisboa, Portugal, 2020.

[5] J. Zhang, H. Luan, M. Sun, F. Zhai, J. Xu, M. Zhang and Y. Liu, "Improving the Transformer Translation Model with Document-Level Context," in *Conference on Empirical Methods in Natural Language Processing*, Brussels, 2018.

[6] S. Maruf, A. F. T. Martins and G. Haffari, "Contextual Neural Model for Translating Bilingual Multi-Speaker Conversations," in *Third Conference on Machine Translation: Research Papers*, Brussels, 2018.

[7] E. d. C. Dalcol and M. Poesio, "Polygloss - A conversational agent for language practice," Gothenburg, Sweden , 2020.

[8] B. McDowell and N. Goodman, "Learning from Omission," in *57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, 2019.

[9] L. Portnoff, E. Gustafson, J. Rollinson and K. Bicknell, "Methods for Language Learning Assessment at Scale: Duolingo Case Study," 2021. [Online]. Available: https://research.duolingo.com/papers/portnoff.edm21.pdf. [Accessed January 2023].

[10] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *NAACL*, Minneapolis, Minnesota, 2019.

[11] E. Kasneci, K. S. S. Kuchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Gunnemann, E. Hullermeier, S. Krusche, G. Kutyniok, T. Michaeli, C. Nerdel and J. Pfeffer, "ChatGPT for Good? On Opportunities and Challenges of Large Language Models for Education," 23 January 2023. [Online]. Available: https://doi.org/10.35542/osf.io/5er8f. [Accessed 1 February 2023].

[12] Y. Kim, H. Lee, J. Shin and K. Jung, "Improving neural question generation using answer

separation," *Proceedings of the AAAI Conference on Artificial Intelligence,* vol. 33, no. 01, pp. 6602-6609, 2019.

[13] B. Settles, G. T. LaFlair and M. Hagiwara, "Machine Learning–Driven Language Assessment," *Transactions of the Association for Computational Linguistics,* vol. 8, pp. 247-263, 2020.

[14] R. Ziai and A. Karnysheva, "Leveraging Task Information in Grammatical Error Correction for Short Answer Assessment through Context-based Reranking," in *10th Workshop on NLP for Computer Assisted Language Learning*, Online, 2021.

[15] O. Topsakal and E. Topsakal, "Framework for A Foreign Language Teaching Software for Children Utilizing AR, Voicebots and ChatGPT (Large Language Models)," *The Journal of Cognitive Systems,* vol. 7, no. 2, pp. 33-38, 2022.

[16] N. Willms and U. Pado, "A Transformer for SAG: What Does it Grade?," in *11th Workshop on NLP for Computer Assisted Language Learning*, Louvain-la-Neuve, Belgium, 2022.

[17] A. H. A. M. Siagian and M. Aritsugi, "Robustness of Word and Character N-gram Combinations in Detecting Deceptive and Truthful Opinions," *ACM Journal of Data and Information Quality,* vol. 12, no. 1, pp. 5:1-5:24, 2020.

[18] N. Y. Hassan, W. H. Gomaa, G. A. Khoriba and M. H. Haggaq, "Credibility Detection in Twitter Using Word N-gram Analysis and Supervised Machine Learning Techniques," *International Journal of Intelligent Engineering and Systems,* vol. 13, no. 1, pp. 291-300, 2020.

[19] T. Kudo, K. Yamamoto and Y. Matsumoto, "Applying Conditional Random Fields to Japanese Morphological Analysis," in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, Barcelona, 2004.

[20] K. Takaoka, S. Hisamoto, N. Kawahara, M. Sakamoto, Y. Uchida and Y. Matsumoto, "Sudachi: a Japanese Tokenizer for Business," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, 2018.

[21] T. Kudo and J. Richardson, "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*, Brussels, 2018.

[22] E. Bataa and J. Wu, "An Investigation of Transfer Learning-Based Sentiment Analysis in Japanese," in *57th Annual Meeting of the Association for Computational Linguistics*, Florence, 2019.

[23] Y. Kikuta, "Bert pretrained model trained on japanese wikipedia articles," GitHub, 2019. [Online]. Available: https://github.com/yoheikikuta/bert-japanese. [Accessed 17 December 2020].

[24] L. Nio and K. Murakami, "Japanese Sentiment Classification Using Bidirectional Long Short-Term Memory Recurrent Neural Network," in *Proceedings of the 24th Annual Meetings of the Association for Natural Language Processing*, Okayama, 2018.

[25] Y. Ke and M. Hagiwara, "Subcharacter Embeddings' Preference on Neural Networks," in *Proceedings of the 25th Annual Meeting of the Association of Natural Language Processing*, Nagoya, 2019.

[26] X. Zhang and Y. LeCun, "Which Encoding is the Best for Text Classification in Chinese, English, Japanese and Korean?," arXiv:1708.02657v2 [cs.CL], 2017.

[27] B. Sun, L. Yang, P. Dong, W. Zhang, J. Dong and C. Young, "Super Characters: A Conversion from Sentiment Classification to Image Classification," in *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Brussels, 2018.

[28] Z. Qu, X. Song, S. Zheng, X. Wang, X. Song and Z. Li, "Improved Bayes Method Based on TF-IDF Feature and Grade Factor Feature for Chinese Information Classification," in *2018 IEEE International Conference on Big Data and Smart Computing*, Shanghai, 2018.

[29] T. Q. Nguyen and D. Chiang, "Transfer Learning across Low-Resource, Related Languages for Neural Machine Translation," in *Eighth International Joint Conference on Natural Language Processing*, Taipei, 2017.

[30] G. Neubig and J. Hu, "Rapid Adaptation of Neural Machine Translation to New Languages," in *Conference on Empirical Methods in Natural Language Processing*, Brussels, 2018.

[31] S. Mayhew, C.-T. Tsai and D. Roth, "Cheap translation for cross-lingual named entity recognition," in *Conference on Empirical Methods in Natural Language Processing*, Copenhagen, 2017.

[32] J. Xie, Z. Yang, G. Neubig, N. A. Smith and J. Carbonell, "Neural Cross-Lingual Named Entity Recognition with Minimal Resources," in *Conference on Empirical Methods in Natural Language Process- ing*, Brussels, 2018.

[33] G. Lample and A. Conneau, "Cross-lingual Language Model Pretraining," in *arXiv preprint arXiv:1901.07291*, 2019.

[34] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzman, E. Grave, M. Ott, L. Zettlemoyer and V. Stoyanov, "Unsupervised Cross-lingual Representation Learning at Scale," in *arXiv preprint arXiv:1911.02116*, 2020.

[35] A. Søgaard, S. Ruder and I. Vulić, "On the limitations of unsupervised bilingual dictionary induction," 2018.

[36] X. Chen, A. H. Awadallah, H. Hassan, W. Wang and C. Cardie, "Multi-Source Cross-Lingual

Model Transfer: Learning What to Share," 2018.

[37] B. Zoph, D. Yuret, J. May and K. Knight, "Transfer Learning for Low-Resource Neural Machine Translation," in *Conference on Empirical Methods in Natural Language Processing*, Austin, Texas , 2016.

[38] W. U. Ahmad, Z. Zhang, X. Ma, E. Hovy, K.-W. Chang and N. Peng, "On Difficulties of Cross-Lingual Transfer with Order Differences: A Case Study on Dependency Parsing," 2019.

[39] A. Lauscher, V. Ravishankar, I. Vulić and G. Glavaš, "From Zero to Hero: On the Limitations of Zero-Shot Cross-Lingual Transfer with Multilingual Transformers," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, 2020.

[40] J. Pfeiffer, I. Vulić, I. Gurevych and S. Ruder, "MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer," 2020.

[41] P. Keung, Y. Lu, G. Szarvas and N. A. Smith, "The Multilingual Amazon Reviews Corpus," 2020.

[42] B. Willie, K. Vincentio, G. I. Winata, S. Cahyawijaya, X. Li, Z. Y. Lim, S. Soleman, R. Mahendra, P. Fung, S. Bahar and A. Purwarianti, "IndoNLU: Benchmark and Resources for Evaluating Indonesian Natural Language Understanding," in *1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, Suzhou, China, 2020.

[43] F. Koto, A. Rahimi, J. H. Lau and T. Baldwin, "IndoLEM and IndoBERT: A Benchmark Dataset and Pre-trained Language Model for Indonesian NLP," in *28th International Conference on Computational Linguistics*, Barcelona, Spain (Online) , 2020.

[44] A. Purwarianti and I. A. P. A. Crisdayanti, "Improving Bi-LSTM Performance for Indonesian Sentiment Analysis Using Paragraph Vector," in *International Conference of Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, Yogyakarta, Indonesia, 2019.

[45] D. Bahdanau, K. Cho and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," in *International Conference on Learning Representations*, San Diego, CA, 2015.

[46] L. Hongzheng, S. Jiu and S. Can, "Revisiting Back-Translation for Low-Resource Machine Translation Between Chinese and Vietnamese," *IEEE Access,* vol. 8, no. 1, pp. 119931-119939, 2020.

[47] A. Mueller and Y. K. Lal, "Sentence-Level Adaptation for Low-Resource Neural Machine Translation," in *Workshop on Technologies for MT of Low Resource Languages*, Dublin, 2019.

[48] M. Yang, R. Wang, K. Chen, M. Utiyama, E. Sumita, M. Zhang and T. Zhao, "Sentence-Level Agreement for Neural Machine Translation," in *Annual Meeting of the Association for Computational Linguistics*, Florence, 2019.

[49] H. Hassan, A. Aue, C. Chen, V. Chowdhary, J. Clark, C. Federmann, X. Huang, M. Junczys-Dowmunt, W. Lewis, M. Li, S. Liu, T.-Y. Liu, R. Luo, A. Menezes, T. Qin, F. Seide, X. Tan, F. Tian and L. Wu, "Achieving Human Parity on Automatic Chinese to English News Translation," in *arXiv preprint arXiv:1803.05567*, 2018.

[50] Unbabel, "Unbabel," 2021. [Online]. Available: https://unbabel.com. [Accessed 28 August 2021].

[51] Kotozna, "Kotozna," 2021. [Online]. Available: https://kotozna.com/. [Accessed 28 August 2021].

[52] Language I/O, "Language I/O," 2021. [Online]. Available: https://languageio.com/. [Accessed 28 August 2021].

[53] M. Rikters, R. Ri, T. Li and T. Nakazawa, "Designing the Business Conversation Corpus," in *Workshop on Asian Translation*, Hongkong, 2019.

[54] T. Kudo, H. Ichikawa and H. Kazawa, "A joint inference of deep case analysis and zero subject generation for Japanese-to-English statistical machine translation," in *Annual Meeting of the Association for Computational Linguistics (Short Papers)*, Baltimore, Maryland, 2014.

[55] H. Taira, K. Sudoh and M. Nagata, "Zero Pronoun Resolution can Improve the Quality of J-E Translation," in *Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation*, Jeju, 2021.

[56] S. Laubli, R. Sennrich and M. Volk, "Has Machine Translation Achieved Human Parity? A Case for Document-level Evaluation," in *Conference on Empirical Methods in Natural Language Processing*, Brussels, 2018.

[57] A. Toral, S. Castilho, K. Hu and A. Way, "Attaining the Unattainable? Reassessing Claims of Human Parity in Neural Machine Translation," in *Proceedings of the Third Conference on Machine Translation: Research Papers*, Brussels, 2018.

[58] A. Lopes, M. A. Farajian, R. Bawden, M. Zhang and A. F. T. Martins, "Document-level Neural MT: A Systematic Comparison," in *22nd Annual Conference of the European Association for Machine Translation*, Lisboa, 2020.

[59] I. Goto, B. Lu, K. P. Chow, E. Sumita and B. K. Tsou, "Overview of the Patent Machine Translation Task at the NTCIR-9 Workshop," in *NTCIR-9 Workshop Meeting*, Tokyo, Japan, 2011.

[60] T. Nakazawa, M. Yaguchi, K. Uchimoto, M. Utiyama, E. Sumita, S. Kurohashi and H. Isahara, "ASPEC: Asian Scientific Paper Excerpt Corpus," in *Tenth International Conference on Language Resources and Evaluation (LREC'16)*, Portorož, Slovenia, 2016.

[61] S. Shimazu, S. Takase, T. Nakazawa and N. Okazaki, " Evaluation Dataset for Zero Pronoun in

Japanese to English Translation," in *Conference on Language Resources and Evaluation (LREC 2020)*, Marseille, 2020.

[62] M. A. Farajian, A. V. Lopes, A. F. T. Martins, S. Maruf and G. Haffari, "Findings of the WMT 2020 Shared Task on Chat Translation," in *5th Conference on Machine Translation (WMT)*, Online, 2020.

[63] C. Federmann, "Appraise: An Open-Source Toolkit for Manual Phrase-Based Evaluation of Translations," in *The Seventh International Conference on Language Resources and Evaluation* , Valletta, Malta, 2010.

[64] S. M. Yimam, I. Gurevych, R. E. d. Castilho and C. Biemann, "WebAnno: A Flexible, Web-based and Visually Supported System for Distributed Annotations," Sofia, Bulgaria , 2013.

[65] J.-C. Klie, M. Bugert, B. Boullosa, R. E. d. Castilho and I. Gurevych, "WebAnno: A Flexible,Web-based and Visually Supported System for Distributed Annotations," Santa Fe, New Mexico , 2018.

[66] M. Shishido, 逆襲の英会話海外旅行編, 旺文社, 1998.

[67] M. Morishita, J. Suzuki and M. Nagata, "JParaCrawl: A Large Scale Web-Based English-Japanese Parallel Corpus," in *The 12th Language Resources and Evaluation Conference*, Marseille, France, 2020.

[68] R. Pryzant, Y. Chung, D. Jurafsky and D. Britz, "JESC: Japanese-English Subtitle Corpus," in *The Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, 2018.

[69] A. Fan, S. Bhosale, H. Schwenk, Z. Ma, A. ElKishky, S. Goyal, M. B. O. Celebi, G. Wenzek, V. Chaudhary, N. Goyal, T. Birch, V. Liptchinsky, S. Edunov, E. Grave, M. Auli and A. Joulin, "Beyond English-Centric Multilingual Machine Translation," *Journal of Machine Learning Research ,* vol. 22, pp. 1-48, 2021.

[70] R. Bawden, R. Sennrich, A. Birch and B. Haddow, "Evaluating Discourse Phenomena in Neural Machine Translation," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, 2018.

[71] F. Hieber, T. Domhan, M. Denkowski, D. Vilar, A. Sokolov, A. Clifton and M. Post, "Sockeye: A Toolkit for Neural Machine Translation," ArXiv e-prints, 2017.

[72] K. Papineni, S. Roukos, T. Ward and W.-J. Zhu, "BLEU: a Method for Automatic Evaluation of Machine Translation," in *40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, 2002.

[73] C. Hardmeier, "Discourse in Statistical Machine Translation: A Survey and a Case Study,"

Discours, Discourse in Statistical Machine Translation, vol. 11, 2012.

[74] E.-L. Kasesniemi, Mobile Messages: Young People and a New Communication Culture, Tampere University Press, 2003.

[75] A. Spagnolli and L. Gamberini, "Interacting via SMS: Practices of social closeness and reciprocation," *British Journal of Social Psychology,* vol. 46, no. 2, pp. 343-364, 2010.

[76] N. Cavus and D. Ibrahim, "m-Learning: An experiment in using SMS to support learning new English language words," *British Journal of Educational Technology,* vol. 40, no. 1, pp. 78-91, 2009.

[77] A. Hayati, A. Jalilifar and A. Mashhadi, "Using Short Message Service (SMS) to teach English idioms to EFL students," *British Journal of Educational Technology,* vol. 44, no. 1, pp. 66-81, 2013.

[78] A. Duff, Translation, 5th Edition, Oxford: Oxford University Press, 1996.

[79] I. Dagilienė, "Translation as a Learning Method in English Language Teaching," *STUDIES ABOUT LANGUAGES,* no. 12, pp. 124-129, 2012.

[80] C. T. Mart, "The Grammar-Translation Method and the Use of Translation to Facilitate Learning in ESL Classes," *Journal of Advances in English Language Teaching,* vol. 1, no. 4, pp. 103-105, 2013.

[81] S. M. Joubran and R. M. Arabiat, "Using Translation in the Framework of Learning a Foreign Language," *Multicultural Education,* vol. 7, no. 8, pp. 61-67, 2021.

[82] R. Rei, C. Stewart, A. C. Farinha and A. Lavie, "COMET: A Neural Framework for MT Evaluation," in *2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, 2020.

[83] S. Castilho, S. Doherty, F. Gaspari and J. Moorkens, "Approaches to Human and Machine Translation Quality Assessment," *Machine Translation: Technologies and Applications book series (MATRA,volume 1),* pp. 9-38, 14 July 2018.

[84] M. Freitag, R. Rei, N. Mathur, C.-k. Lo, C. Stewart, G. Foster, A. Lavie and O. Bojar, "Results of the WMT21 Metrics Shared Task: Evaluating Metrics with Expert-based Human Evaluations on TED and News Domain," in *Sixth Conference on Machine Translation*, Online, 2021.

[85] L. Han, "Machine translation evaluation resources and methods: A survey," ArXiv: Computation and language, 2016.

[86] B. Chen and C. Cherry, "A Systematic Comparison of Smoothing Techniques for Sentence-Level BLEU," in *Ninth Workshop on Statistical Machine Translation*, Baltimore, Maryland, USA, 2014.

[87] E. Reiter, "A Structured Review of the Validity of BLEU," *Computational Linguistics,* vol. 44,

no. 3, p. 393–401 , 2018.

[88] M. Hanna and O. Bojar, "A Fine-Grained Analysis of BERTScore," Online, 2021.

[89] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger and Y. Artzi, "BERTScore: Evaluating Text Generation with BERT," Online, 2020.

[90] N. Mathur, T. Baldwin and T. Cohn, "Tangled up in BLEU: Reevaluating the Evaluation of Automatic Machine Translation Evaluation Metrics," Online, 2020.

[91] M. Snover, B. Dorr, R. Schwartz, L. Micciulla and J. Makhoul, "A Study of Translation Edit Rate with Targeted Human Annotation," Cambridge, Massachusetts, USA , 2006.

[92] S. Banerjee and A. Lavie, "METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments," Ann Arbor, Michigan , 2005.

[93] M. Popović, "chrF: character n-gram F-score for automatic MT evaluation," Lisbon, Portugal, 2015.

[94] OpenAI, "OpenAI," 30 November 2022. [Online]. Available: https://openai.com/blog/chatgpt/. [Accessed December 2022].

Appendix

This section shows snippets/parts of codes which were created during the experiments in each chapter described above. However, please note that only experimental codes that can be contained in a file, not all, are listed below due to the limited space. The list below includes snippets of codes for:

- Implementation and Comparison of Various Japanese Tokenizers (Chapter II)
- Fine-tuning XLM-RoBERTa for Cross-lingual Classification (Chapter II)
- Training a Japanese→English NMT Model (Chapter III)
- Providing an API Endpoint for Japanese→English Translation (Chapter IV)
- Evaluation and Comparison of MT Evaluation Metrics (Chapter V)

A. Implementation and Comparison Various Japanese Tokenizers (Chapter II)

```
# %% [markdown]
…
# ## Train a SentencePiece Model from Train Data (vocab_size=32000)
# %%
start = time.time()
spm.SentencePieceTrainer.train(input='../sampled_binary_train.csv', model_prefix='sp-model-32000-340k',
vocab_size=32000)
end = time.time()
print('Time to train a SP model w/ 32000 vocab_size, from 340k train data:', end-start)

# %% [markdown]
# ## Tokenization Feature

# %%
# mecab w/ unidic-lite
wakati = MeCab.Tagger("-Owakati")

#sudachi
sudachi = dictionary.Dictionary().create()
mode = tokenizer.Tokenizer.SplitMode.B

#sentencepiece
sp = spm.SentencePieceProcessor(model_file='./sp-model-32000-340k.model')
```

```
# %% [markdown]
# ## TF-IDF Vectorizer

# %%
def tokenize_sp(text):
        tokenized = sp.encode(text, out_type=str)
        return tokenized

# %%
def tokenize_mecab(text):
        tokenized = wakati.parse(text).split()
        return tokenized

# %%
def tokenize_sudachi(text):
        tokenized = [m.surface() for m in sudachi.tokenize(text, mode)]
        return tokenized

# %%
X_train, y_train = df_train[2], df_train[0]
print("Total train samples: ", len(X_train))

# %%
tfidfVect_mecab = TfidfVectorizer(tokenizer=tokenize_mecab)
tfidfVect_sudachi = TfidfVectorizer(tokenizer=tokenize_sudachi)
tfidfVect_sp = TfidfVectorizer(tokenizer=tokenize_sp)

# %%
start = time.time()
X_train_tfidf_mecab = tfidfVect_mecab.fit_transform(X_train)
end = time.time()
print("TFIDF Vect time (MeCab): ", end-start)

start = time.time()
X_train_tfidf_sudachi = tfidfVect_sudachi.fit_transform(X_train)
```

```
end = time.time()
print("TFIDF Vect time (Sudachi): ", end-start)

start = time.time()
X_train_tfidf_sp = tfidfVect_sp.fit_transform(X_train)
end = time.time()
print("TFIDF Vect time (SentencePiece): ", end-start)

# %% [markdown]
# ## Model Training

# %%
start = time.time()
clf_lr_mecab = LogisticRegression(random_state=0).fit(X_train_tfidf_mecab, y_train)
end = time.time()
print("Training time (LR-MeCab): ", end-start)

start = time.time()
clf_lr_sudachi = LogisticRegression(random_state=0).fit(X_train_tfidf_sudachi, y_train)
end = time.time()
print("Training time (LR-Sudachi): ", end-start)

start = time.time()
clf_lr_sp = LogisticRegression(random_state=0).fit(X_train_tfidf_sp, y_train)
end = time.time()
print("Training time (LR-SP): ", end-start)

start = time.time()
clf_mnb_mecab = MultinomialNB().fit(X_train_tfidf_mecab, y_train)
end = time.time()
print("Training time (MNB-MeCab): ", end-start)

start = time.time()
clf_mnb_sudachi = MultinomialNB().fit(X_train_tfidf_sudachi, y_train)
end = time.time()
print("Training time (MNB-Sudachi): ", end-start)
```

```
start = time.time()
clf_mnb_sp = MultinomialNB().fit(X_train_tfidf_sp, y_train)
end = time.time()
print("Training time (MNB-SP): ", end-start)

# %% [markdown]
# ## Evaluation (Error Rate = 1 - accuracy)
# Predictions on test set
X_test, y_test = df_test[2], df_test[0]
print("Total test samples: ", len(X_test))

X_test_tfidf = tfidfVect_mecab.transform(X_test)
predicted = clf_lr_mecab.predict(X_test_tfidf)
print("Error rate (LR-MeCab)", (1-np.mean(predicted == y_test))*100)

X_test_tfidf = tfidfVect_sudachi.transform(X_test)
predicted = clf_lr_sudachi.predict(X_test_tfidf)
print("Error rate (LR-Sudachi)", (1-np.mean(predicted == y_test))*100)

X_test_tfidf = tfidfVect_sp.transform(X_test)
predicted = clf_lr_sp.predict(X_test_tfidf)
print("Error rate (LR-SP)", (1-np.mean(predicted == y_test))*100)

X_test_tfidf = tfidfVect_mecab.transform(X_test)
predicted = clf_mnb_mecab.predict(X_test_tfidf)
print("Error rate (MNB-MeCab)", (1-np.mean(predicted == y_test))*100)

X_test_tfidf = tfidfVect_sudachi.transform(X_test)
predicted = clf_mnb_sudachi.predict(X_test_tfidf)
print("Error rate (MNB-Sudachi)", (1-np.mean(predicted == y_test))*100)

X_test_tfidf = tfidfVect_sp.transform(X_test)
predicted = clf_mnb_sp.predict(X_test_tfidf)
print("Error rate (MNB-SP)", (1-np.mean(predicted == y_test))*100)
```

```
# %% [markdown]
# ## Grid Search CV -- Logistics Regression Parameter Tuning

# %%
# Grid Search CV => Logistics Regression + SentencePiece

from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import GridSearchCV

start = time.time()
# set parameters
model = LogisticRegression(random_state=0)
solvers = ['newton-cg', 'lbfgs', 'liblinear']
penalty = ['l2']
c_values = [100, 10, 1.0, 0.1, 0.01]
# define grid search
grid = dict(solver=solvers,penalty=penalty,C=c_values)
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
grid_search = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1, cv=cv,
scoring='accuracy',error_score=0)
grid_result = grid_search.fit(X_train_tfidf_sp, y_train)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
end = time.time()
print("Training time (LR-SP): ", end-start)

# %% [markdown]
# ## Model Training and Evaluation

# %% [markdown]
# ### Logistics Regression(random_state=0, C=10, penalty=l2, solver lbfgs -- SentencePiece
# %%
start = time.time()
clf_lr_sp = LogisticRegression(random_state=0, C=10, penalty='l2', solver='lbfgs').fit(X_train_tfidf_sp,
y_train)
end = time.time()
print("Training time (LR-SP): ", end-start, " seconds.")

predicted = clf_lr_sp.predict(X_train_tfidf_sp)
print("Train Error rate (LR-SP)", (1-np.mean(predicted == y_train))*100)

X_test, y_test = df_test[2], df_test[0]
print("Total test samples: ", len(X_test))

X_test_tfidf = tfidfVect_sp.transform(X_test)
predicted = clf_lr_sp.predict(X_test_tfidf)
print("Test Error rate (LR-SP)", (1-np.mean(predicted == y_test))*100)
```

B. Fine-tuning XLM-RoBERTa for Cross-lingual Classification (Chapter II)

```
# %%
…
# %%
from google.colab import drive
drive.mount('/content/drive')

# %%
# Get the GPU device name.
device_name = tf.test.gpu_device_name()

# The device name should look like the following:
if device_name == '/device:GPU:0':
        print('Found GPU at: {}'.format(device_name))
else:
        raise SystemError('GPU device not found')

# If there's a GPU available...
```

```
if torch.cuda.is_available():

        # Tell PyTorch to use the GPU.
        device = torch.device("cuda")

        print('There are %d GPU(s) available.' % torch.cuda.device_count())

        print('We will use the GPU:', torch.cuda.get_device_name(0))

# If not...
else:
        print('No GPU available, using the CPU instead.')
        device = torch.device("cpu")

# %% [markdown]
# # Fine-Tuning XLM-R for Binary Sentiment Classification
….
# %% [markdown]
# ## XLM-RoBERTa Tokenizer

# %%
# Download the tokenizer for the XLM-Robert `base` model.
tokenizer = XLMRobertaTokenizer.from_pretrained("xlm-roberta-base")
…

# %% [markdown]
# ## Convert Sentences to Input IDs
# %%
input_ids = []

for sent in reviews:
        encoded_sent = tokenizer.encode(
                                        sent,
                                        add_special_tokens = True,
                                        max_length = 128,
                                        truncation=True,
                                    )
        input_ids.append(encoded_sent)

# Print sentence 0, now as a list of IDs.
print('Original: ', reviews[0])
print('Token IDs:', input_ids[0])

# %%
print('Max sentence length: ', max([len(sen) for sen in input_ids]))

# %% [markdown]
# %%
MAX_LEN = 64
print('¥nPadding/truncating all sentences to %d values...' % MAX_LEN)
print('¥nPadding token: "{:}", ID: {:}'.format(tokenizer.pad_token, tokenizer.pad_token_id))

input_ids = pad_sequences(input_ids, maxlen=MAX_LEN, dtype="long",
                                                    value=0, truncating="post", padding="post")

print('¥nDone.')

# %% [markdown]
# %%
attention_masks = []

for sent in input_ids:
        att_mask = [int(token_id > 0) for token_id in sent]
        attention_masks.append(att_mask)

# %% [markdown]
# ## Train & Validation Split

# %%
# Use 90% for training and 10% for validation.
train_inputs, validation_inputs, train_labels, validation_labels = train_test_split(input_ids, sentiments,
                                                                              random_state=0,
test_size=0.1)
train_masks, validation_masks, _, _ = train_test_split(attention_masks, sentiments, random_state=0,
test_size=0.1)

train_inputs = torch.tensor(train_inputs)
validation_inputs = torch.tensor(validation_inputs)

train_labels = torch.tensor(train_labels)
validation_labels = torch.tensor(validation_labels)

train_masks = torch.tensor(train_masks)
validation_masks = torch.tensor(validation_masks)

batch_size = 32

# DataLoader for our training set.
train_data = TensorDataset(train_inputs, train_masks, train_labels)
train_sampler = RandomSampler(train_data)
train_dataloader = DataLoader(train_data, sampler=train_sampler, batch_size=batch_size)

# DataLoader for our validation set.
validation_data = TensorDataset(validation_inputs, validation_masks, validation_labels)
validation_sampler = SequentialSampler(validation_data)
validation_dataloader = DataLoader(validation_data, sampler=validation_sampler, batch_size=batch_size)

# %% [markdown]
# ## Load the XLM-R Pre-trained Model

# %%
model = XLMRobertaForSequenceClassification.from_pretrained(
    "xlm-roberta-base", # Use the 12-layer BERT model, with an uncased vocab.
    num_labels = 2, # 2 for binary classification.
    output_attentions = False, # Whether the model returns attentions weights.
    output_hidden_states = False, # Whether the model returns all hidden-states.
)
```

```
model.cuda()
…
# %% [markdown]
# ## Optimizer & Learning Rate Scheduler
# Batch size: 32
# Learning rate (AdamW): 2e-5
# Number of epochs: 4
# %%
optimizer = AdamW(model.parameters(),
                  lr = 2e-5,
                  eps = 1e-8
                )
# %%
from transformers import get_linear_schedule_with_warmup
epochs = 4
total_steps = len(train_dataloader) * epochs

scheduler = get_linear_schedule_with_warmup(optimizer,
                                            num_warmup_steps = 0,
                                            num_training_steps = total_steps)

# %%
def flat_accuracy(preds, labels):
    pred_flat = np.argmax(preds, axis=1).flatten()
    labels_flat = labels.flatten()
    return np.sum(pred_flat == labels_flat) / len(labels_flat)

# %%
#Helper function for formatting elapsed times.
def format_time(elapsed):
    '''
    Takes a time in seconds and returns a string hh:mm:ss
    '''
    # Round to the nearest second.
    elapsed_rounded = int(round((elapsed)))
```

```
        # Format as hh:mm:ss
        return str(datetime.timedelta(seconds=elapsed_rounded))

# %%
seed_val = 42

random.seed(seed_val)
np.random.seed(seed_val)
torch.manual_seed(seed_val)
torch.cuda.manual_seed_all(seed_val)

loss_values = []

for epoch_i in range(0, epochs):
        print("")
        print('======== Epoch {:} / {:} ========'.format(epoch_i + 1, epochs))
        print('Training...')
        t0 = time.time()
        total_loss = 0
        model.train()

        for step, batch in enumerate(train_dataloader):
                if step % 40 == 0 and not step == 0:
                        elapsed = format_time(time.time() - t0)
                        print('    Batch {:>5,}    of    {:>5,}.        Elapsed: {:}.'.format(step, len(train_dataloader),
elapsed))
                b_input_ids = batch[0].to(device)
                b_input_mask = batch[1].to(device)
                b_labels = batch[2].to(device)

                model.zero_grad()
                outputs = model(b_input_ids,
                                        token_type_ids=None,
                                        attention_mask=b_input_mask,
                                        labels=b_labels)
        loss = outputs[0]
        total_loss += loss.item()
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
        optimizer.step()
        scheduler.step()

    avg_train_loss = total_loss / len(train_dataloader)
    loss_values.append(avg_train_loss)

    print("")
    print("    Average training loss: {0:.2f}".format(avg_train_loss))
    print("    Training epoch took: {:}".format(format_time(time.time() - t0)))

    # Validation
    t0 = time.time()
    model.eval()
    eval_loss, eval_accuracy = 0, 0
    nb_eval_steps, nb_eval_examples = 0, 0

    # Evaluate data for one epoch
    for batch in validation_dataloader:
        batch = tuple(t.to(device) for t in batch)
        b_input_ids, b_input_mask, b_labels = batch
        with torch.no_grad():
            outputs = model(b_input_ids,
                            token_type_ids=None,
                            attention_mask=b_input_mask)
        logits = outputs[0]
        logits = logits.detach().cpu().numpy()
        label_ids = b_labels.to('cpu').numpy()
        tmp_eval_accuracy = flat_accuracy(logits, label_ids)
        eval_accuracy += tmp_eval_accuracy
        nb_eval_steps += 1
        print("    Accuracy: {0:.2f}".format(eval_accuracy/nb_eval_steps))
        print("    Validation took: {:}".format(format_time(time.time() - t0)))
print("Training complete!")

# Save a trained model, configuration and tokenizer using `save_pretrained()`.
# This can be reloaded using `from_pretrained()`
model_to_save = model.module if hasattr(model, 'module') else model
model_to_save.save_pretrained(output_dir)
tokenizer.save_pretrained(output_dir)

# %%
!ls -l --block-size=K ./model_save/

# %%
!ls -l --block-size=M ./model_save/pytorch_model.bin

# %%
# Copy the model files to a directory in your Google Drive.
!cp -r ./model_save/ "./models/enja-binary-model/"
…
```

C. Training a Japanese→English NMT Model (Chapter III)

```
…
en_tokenizer = spm.SentencePieceProcessor(model_file='spm.en.nopretok.model')
ja_tokenizer = spm.SentencePieceProcessor(model_file='spm.ja.nopretok.model')

def build_vocab(sentences, tokenizer):
    counter = Counter()
    for sentence in sentences:
        counter.update(tokenizer.encode(sentence, out_type=str))
    return Vocab(counter, specials=['<unk>', '<pad>', '<bos>', '<eos>'])
ja_vocab = build_vocab(trainja, ja_tokenizer)
en_vocab = build_vocab(trainen, en_tokenizer)

def data_process(ja, en):
    data = []
    for (raw_ja, raw_en) in zip(ja, en):
        ja_tensor_ = torch.tensor([ja_vocab[token] for token in ja_tokenizer.encode(raw_ja.rstrip("¥n"),
out_type=str)],
                                                    dtype=torch.long)
        en_tensor_ = torch.tensor([en_vocab[token] for token in en_tokenizer.encode(raw_en.rstrip("¥n"),
out_type=str)],
                                                    dtype=torch.long)
        data.append((ja_tensor_, en_tensor_))
    return data
train_data = data_process(trainja, trainen)

BATCH_SIZE = 16
PAD_IDX = ja_vocab['<pad>']
BOS_IDX = ja_vocab['<bos>']
EOS_IDX = ja_vocab['<eos>']
def generate_batch(data_batch):
    ja_batch, en_batch = [], []
    for (ja_item, en_item) in data_batch:
        ja_batch.append(torch.cat([torch.tensor([BOS_IDX]), ja_item, torch.tensor([EOS_IDX])], dim=0))
        en_batch.append(torch.cat([torch.tensor([BOS_IDX]), en_item, torch.tensor([EOS_IDX])], dim=0))
    ja_batch = pad_sequence(ja_batch, padding_value=PAD_IDX)
    en_batch = pad_sequence(en_batch, padding_value=PAD_IDX)
    return ja_batch, en_batch
train_iter = DataLoader(train_data, batch_size=BATCH_SIZE,
                                            shuffle=True, collate_fn=generate_batch)

from torch.nn import (TransformerEncoder, TransformerDecoder,
                                    TransformerEncoderLayer, TransformerDecoderLayer)


class Seq2SeqTransformer(nn.Module):
        def __init__(self, num_encoder_layers: int, num_decoder_layers: int,
                                emb_size: int, src_vocab_size: int, tgt_vocab_size: int,
                                dim_feedforward:int = 512, dropout:float = 0.1):
                super(Seq2SeqTransformer, self).__init__()
                encoder_layer = TransformerEncoderLayer(d_model=emb_size, nhead=NHEAD,
                                                                        dim_feedforward=dim_feedforward)
                self.transformer_encoder = TransformerEncoder(encoder_layer,
num_layers=num_encoder_layers)
                decoder_layer = TransformerDecoderLayer(d_model=emb_size, nhead=NHEAD,
                                                        dim_feedforward=dim_feedforward)
                self.transformer_decoder = TransformerDecoder(decoder_layer,
num_layers=num_decoder_layers)

                self.generator = nn.Linear(emb_size, tgt_vocab_size)
                self.src_tok_emb = TokenEmbedding(src_vocab_size, emb_size)
                self.tgt_tok_emb = TokenEmbedding(tgt_vocab_size, emb_size)
                self.positional_encoding = PositionalEncoding(emb_size, dropout=dropout)

        def forward(self, src: Tensor, trg: Tensor, src_mask: Tensor,
                    tgt_mask: Tensor, src_padding_mask: Tensor,
                    tgt_padding_mask: Tensor, memory_key_padding_mask: Tensor):
                src_emb = self.positional_encoding(self.src_tok_emb(src))
                tgt_emb = self.positional_encoding(self.tgt_tok_emb(trg))
                memory = self.transformer_encoder(src_emb, src_mask, src_padding_mask)
                outs = self.transformer_decoder(tgt_emb, memory, tgt_mask, None,
                                                tgt_padding_mask, memory_key_padding_mask)
                return self.generator(outs)

        def encode(self, src: Tensor, src_mask: Tensor):
                return self.transformer_encoder(self.positional_encoding(
                                    self.src_tok_emb(src)), src_mask)

        def decode(self, tgt: Tensor, memory: Tensor, tgt_mask: Tensor):
                return self.transformer_decoder(self.positional_encoding(
                                  self.tgt_tok_emb(tgt)), memory,
                                  tgt_mask)

class PositionalEncoding(nn.Module):
        def __init__(self, emb_size: int, dropout, maxlen: int = 5000):
                super(PositionalEncoding, self).__init__()
                den = torch.exp(- torch.arange(0, emb_size, 2) * math.log(10000) / emb_size)
                pos = torch.arange(0, maxlen).reshape(maxlen, 1)
                pos_embedding = torch.zeros((maxlen, emb_size))
                pos_embedding[:, 0::2] = torch.sin(pos * den)
                pos_embedding[:, 1::2] = torch.cos(pos * den)
                pos_embedding = pos_embedding.unsqueeze(-2)

                self.dropout = nn.Dropout(dropout)
                self.register_buffer('pos_embedding', pos_embedding)

        def forward(self, token_embedding: Tensor):
                return self.dropout(token_embedding +
                                    self.pos_embedding[:token_embedding.size(0),:])

class TokenEmbedding(nn.Module):
        def __init__(self, vocab_size: int, emb_size):
                super(TokenEmbedding, self).__init__()
                self.embedding = nn.Embedding(vocab_size, emb_size)
                self.emb_size = emb_size
        def forward(self, tokens: Tensor):
                return self.embedding(tokens.long()) * math.sqrt(self.emb_size)

def generate_square_subsequent_mask(sz):
        mask = (torch.triu(torch.ones((sz, sz), device=DEVICE)) == 1).transpose(0, 1)
        mask = mask.float().masked_fill(mask == 0, float('-inf')).masked_fill(mask == 1, float(0.0))
        return mask

def create_mask(src, tgt):
  src_seq_len = src.shape[0]
  tgt_seq_len = tgt.shape[0]

  tgt_mask = generate_square_subsequent_mask(tgt_seq_len)
  src_mask = torch.zeros((src_seq_len, src_seq_len), device=DEVICE).type(torch.bool)

  src_padding_mask = (src == PAD_IDX).transpose(0, 1)
  tgt_padding_mask = (tgt == PAD_IDX).transpose(0, 1)
  return src_mask, tgt_mask, src_padding_mask, tgt_padding_mask
```

```
SRC_VOCAB_SIZE = len(ja_vocab)
TGT_VOCAB_SIZE = len(en_vocab)
EMB_SIZE = 512
NHEAD = 8
FFN_HID_DIM = 512
BATCH_SIZE = 16
NUM_ENCODER_LAYERS = 3
NUM_DECODER_LAYERS = 3
NUM_EPOCHS = 16
transformer = Seq2SeqTransformer(NUM_ENCODER_LAYERS, NUM_DECODER_LAYERS,
                                 EMB_SIZE, SRC_VOCAB_SIZE, TGT_VOCAB_SIZE,
                                 FFN_HID_DIM)

for p in transformer.parameters():
    if p.dim() > 1:
        nn.init.xavier_uniform_(p)

transformer = transformer.to(device)

loss_fn = torch.nn.CrossEntropyLoss(ignore_index=PAD_IDX)

optimizer = torch.optim.Adam(
    transformer.parameters(), lr=0.0001, betas=(0.9, 0.98), eps=1e-9
)
def train_epoch(model, train_iter, optimizer):
  model.train()
  losses = 0
  for idx, (src, tgt) in enumerate(train_iter):
      src = src.to(device)
      tgt = tgt.to(device)

      tgt_input = tgt[:-1, :]

      src_mask, tgt_mask, src_padding_mask, tgt_padding_mask = create_mask(src, tgt_input)

      logits = model(src, tgt_input, src_mask, tgt_mask,
                                                      src_padding_mask, tgt_padding_mask, src_padding_mask)

            optimizer.zero_grad()

            tgt_out = tgt[1:,:]
            loss = loss_fn(logits.reshape(-1, logits.shape[-1]), tgt_out.reshape(-1))
            loss.backward()

            optimizer.step()
            losses += loss.item()
    return losses / len(train_iter)


def evaluate(model, val_iter):
    model.eval()
    losses = 0
    for idx, (src, tgt) in (enumerate(valid_iter)):
        src = src.to(device)
        tgt = tgt.to(device)

        tgt_input = tgt[:-1, :]

        src_mask, tgt_mask, src_padding_mask, tgt_padding_mask = create_mask(src, tgt_input)

        logits = model(src, tgt_input, src_mask, tgt_mask,
                                                      src_padding_mask, tgt_padding_mask, src_padding_mask)
        tgt_out = tgt[1:,:]
        loss = loss_fn(logits.reshape(-1, logits.shape[-1]), tgt_out.reshape(-1))
        losses += loss.item()
    return losses / len(val_iter)

for epoch in range(1, NUM_EPOCHS+1):
    start_time = time.time()
    train_loss = train_epoch(transformer, train_iter, optimizer)
    end_time = time.time()
    print((f"Epoch: {epoch}, Train loss: {train_loss:.3f}, "
            f"Epoch time = {(end_time - start_time):.3f}s"))
```

D. Providing an API Endpoint for Japanese→English Translation (Chapter IV)

```
…
en_tok = spm.SentencePieceProcessor(
        model_file='enja_spm_models/spm.en.nopretok.model')
ja_tok = spm.SentencePieceProcessor(
        model_file='enja_spm_models/spm.ja.nopretok.model')

# open a file, where you ant to store the data
file = open('en_vocab_all.pkl', 'rb')
# load information from file
en_voc = pickle.load(file)
# close the file
file.close()

# open a file, where you ant to store the data
file = open('ja_vocab_all.pkl', 'rb')
# load information from file
ja_voc = pickle.load(file)
# close the file
file.close()

PAD_IDX = ja_voc['<pad>']
BOS_IDX = ja_voc['<bos>']
EOS_IDX = ja_voc['<eos>']

SRC_VOCAB_SIZE = len(ja_voc)
TGT_VOCAB_SIZE = len(en_voc)
EMB_SIZE = 512
NHEAD = 8
FFN_HID_DIM = 512
BATCH_SIZE = 16    # reduce this to prevent cuda out of memory?
NUM_ENCODER_LAYERS = 3
NUM_DECODER_LAYERS = 3
NUM_EPOCHS = 16
```

```
class Seq2SeqTransformer(nn.Module):
    def __init__(self, num_encoder_layers: int, num_decoder_layers: int,
                 emb_size: int, src_vocab_size: int, tgt_vocab_size: int,
                 dim_feedforward: int = 512, dropout: float = 0.1):
        super(Seq2SeqTransformer, self).__init__()
        encoder_layer = TransformerEncoderLayer(d_model=emb_size, nhead=NHEAD,
                                                dim_feedforward=dim_feedforward)
        self.transformer_encoder = TransformerEncoder(
            encoder_layer, num_layers=num_encoder_layers)
        decoder_layer = TransformerDecoderLayer(d_model=emb_size, nhead=NHEAD,
                                                dim_feedforward=dim_feedforward)
        self.transformer_decoder = TransformerDecoder(
            decoder_layer, num_layers=num_decoder_layers)

        self.generator = nn.Linear(emb_size, tgt_vocab_size)
        self.src_tok_emb = TokenEmbedding(src_vocab_size, emb_size)
        self.tgt_tok_emb = TokenEmbedding(tgt_vocab_size, emb_size)
        self.positional_encoding = PositionalEncoding(
            emb_size, dropout=dropout)

    def forward(self, src: Tensor, trg: Tensor, src_mask: Tensor,
                tgt_mask: Tensor, src_padding_mask: Tensor,
                tgt_padding_mask: Tensor, memory_key_padding_mask: Tensor):
        src_emb = self.positional_encoding(self.src_tok_emb(src))
        tgt_emb = self.positional_encoding(self.tgt_tok_emb(trg))
        memory = self.transformer_encoder(src_emb, src_mask, src_padding_mask)
        outs = self.transformer_decoder(tgt_emb, memory, tgt_mask, None,
                                        tgt_padding_mask, memory_key_padding_mask)
        return self.generator(outs)

    def encode(self, src: Tensor, src_mask: Tensor):
        return self.transformer_encoder(self.positional_encoding(
            self.src_tok_emb(src)), src_mask)

    def decode(self, tgt: Tensor, memory: Tensor, tgt_mask: Tensor):
        return self.transformer_decoder(self.positional_encoding(
                    self.tgt_tok_emb(tgt)), memory,
                    tgt_mask)

class PositionalEncoding(nn.Module):
    def __init__(self, emb_size: int, dropout, maxlen: int = 5000):
        super(PositionalEncoding, self).__init__()
        den = torch.exp(- torch.arange(0, emb_size, 2)
                            * math.log(10000) / emb_size)
        pos = torch.arange(0, maxlen).reshape(maxlen, 1)
        pos_embedding = torch.zeros((maxlen, emb_size))
        pos_embedding[:, 0::2] = torch.sin(pos * den)
        pos_embedding[:, 1::2] = torch.cos(pos * den)
        pos_embedding = pos_embedding.unsqueeze(-2)

        self.dropout = nn.Dropout(dropout)
        self.register_buffer('pos_embedding', pos_embedding)

    def forward(self, token_embedding: Tensor):
        return self.dropout(token_embedding +
                               self.pos_embedding[:token_embedding.size(0), :])

class TokenEmbedding(nn.Module):
    def __init__(self, vocab_size: int, emb_size):
        super(TokenEmbedding, self).__init__()
        self.embedding = nn.Embedding(vocab_size, emb_size)
        self.emb_size = emb_size

    def forward(self, tokens: Tensor):
        return self.embedding(tokens.long()) * math.sqrt(self.emb_size)

def generate_square_subsequent_mask(sz):
    mask = (torch.triu(torch.ones((sz, sz), device=DEVICE)) == 1).transpose(0, 1)
    mask = mask.float().masked_fill(mask == 0, float(
        '-inf')).masked_fill(mask == 1, float(0.0))
    return mask
```

```
def create_mask(src, tgt):
    src_seq_len = src.shape[0]
    tgt_seq_len = tgt.shape[0]

    tgt_mask = generate_square_subsequent_mask(tgt_seq_len)
    src_mask = torch.zeros((src_seq_len, src_seq_len),
                           device=DEVICE).type(torch.bool)

    src_padding_mask = (src == PAD_IDX).transpose(0, 1)
    tgt_padding_mask = (tgt == PAD_IDX).transpose(0, 1)
    return src_mask, tgt_mask, src_padding_mask, tgt_padding_mask

def greedy_decode(model, src, src_mask, max_len, start_symbol):
    src = src.to(device)
    src_mask = src_mask.to(device)

    memory = model.encode(src, src_mask)
    ys = torch.ones(1, 1).fill_(start_symbol).type(torch.long).to(device)
    for i in range(max_len-1):
        memory = memory.to(device)
        memory_mask = torch.zeros(ys.shape[0], memory.shape[0]).to(
            device).type(torch.bool)
        tgt_mask = (generate_square_subsequent_mask(ys.size(0))
                    .type(torch.bool)).to(device)
        out = model.decode(ys, memory, tgt_mask)
        out = out.transpose(0, 1)
        prob = model.generator(out[:, -1])
        _, next_word = torch.max(prob, dim=1)
        next_word = next_word.item()

        ys = torch.cat([ys,
                        torch.ones(1, 1).type_as(src.data).fill_(next_word)], dim=0)
        if next_word == EOS_IDX:
            break
    return ys
```

```
def translate(model, src, src_vocab, tgt_vocab, src_tokenizer, tgt_tokenizer):
    model.eval()
    tokens = [BOS_IDX] + [src_vocab.stoi[tok]
                          for tok in src_tokenizer.encode(src, out_type=str)] + [EOS_IDX]
    num_tokens = len(tokens)
    src = (torch.LongTensor(tokens).reshape(num_tokens, 1))
    src_mask = (torch.zeros(num_tokens, num_tokens)).type(torch.bool)
    tgt_tokens = greedy_decode(
        model,    src, src_mask, max_len=num_tokens + 5, start_symbol=BOS_IDX).flatten()
    return tgt_tokenizer.decode([tgt_vocab.itos[tok] for tok in tgt_tokens]).replace("<bos> ",
"").replace("<eos>", "")

mdl = Seq2SeqTransformer(NUM_ENCODER_LAYERS, NUM_DECODER_LAYERS,
                         EMB_SIZE, SRC_VOCAB_SIZE, TGT_VOCAB_SIZE,
                         FFN_HID_DIM)

mdl.load_state_dict(torch.load(
    'inference_model_jpara_jesc_bsd_backtranslated', map_location=torch.device('cpu')))
mdl = mdl.to(device)

app = Flask(__name__)
CORS(app)

@app.route("/translate", methods=['POST'])
def translateJa():
    record = json.loads(request.data)
    translated = translate(mdl, record['chat'], ja_voc, en_voc, ja_tok, en_tok)
    return jsonify({'translated': translated})

if __name__ == '__main__':
    port = int(os.environ.get("PORT", 5000))
    app.run(host='0.0.0.0', port=port)
```

E. Evaluation and Comparison of MT Evaluation Metrics (Chapter V)

```
…
"""# Load Parallel Data w/ Translations"""
```

```
import pandas as pd

df = pd.read_csv('results/results_with_da_scores_our_model.tsv', sep='¥t')
refs, preds, srcs = df['Target_En'].tolist(), df['Translation_En_Ours'].tolist(), df['Source_Ja'].tolist()

refs2d = []
for ref in refs:
        refs2d.append([ref])

len(refs), len(preds), len(refs2d)

df_all = pd.read_json('bsd_dev.json')
print(df_all['tag'].unique())
df_convs = pd.concat([pd.json_normalize(df_all['conversation'][i]) for i in
range(len(df_all))],ignore_index=True)
english = df_convs["en_sentence"].values.tolist()
japanese = df_convs["ja_sentence"].values.tolist()
len(english)

df_all

"""# Calculate [BERTScore](https://huggingface.co/spaces/evaluate-metric/bertscore)"""

from google.colab import output
output.enable_custom_widget_manager()

bertscore = evaluate.load("bertscore")
…

with open(b"results/bertscore_results_our_model.obj","rb") as file:
    bertscore_results = pickle.load(file)

len(bertscore_results["precision"]), len(bertscore_results["recall"]), len(bertscore_results["f1"])

df['BertScore_Prec'] = bertscore_results["precision"]
```

```
df['BertScore_Recall'] = bertscore_results["recall"]
df['BertScore_F1'] =bertscore_results["f1"]
df

"""# Calculate [COMET](https://huggingface.co/spaces/evaluate-metric/comet)"""

comet_metric = evaluate.load('comet')
…
with open(b"results/comet_results_our_model.obj","rb") as file:
    comet_results = pickle.load(file)

df['Comet_Score'] = comet_results['scores']

"""# Calculate Baseline Metrics

## BLEU
"""

from google.colab import output
output.enable_custom_widget_manager()

bleus = []
bleu = evaluate.load("bleu")
st = time.time()
for idx, pred in enumerate(preds):
    bleus.append(bleu.compute(predictions=[pred], references=[refs2d[idx]])['bleu'])
en = time.time()
print(en-st, 'seconds')
print(len(bleus))

"""## chrF"""

from google.colab import output
output.enable_custom_widget_manager()

chrfs = []
```

```
chrf = evaluate.load("chrf")
st = time.time()
for idx, pred in enumerate(preds):
    chrfs.append(chrf.compute(predictions=[pred], references=[refs2d[idx]])['score'])
en = time.time()
print(en-st, 'seconds')
print(len(chrfs))

"""## TER"""

from google.colab import output

output.enable_custom_widget_manager()
ters = []
ter = evaluate.load("ter")
st = time.time()
for idx, pred in enumerate(preds):
    ters.append(ter.compute(predictions=[preds[idx]], references=[refs2d[idx]])['score'])
en = time.time()
print(en-st, 'seconds')
print(len(ters))

"""## METEOR"""

from google.colab import output
output.enable_custom_widget_manager()

meteors = []
meteor = evaluate.load('meteor')
st = time.time()
for idx, pred in enumerate(preds):
    meteors.append(meteor.compute(predictions=[preds[idx]], references=[refs2d[idx]])['meteor'])
en = time.time()
print(en-st, 'seconds')
print(len(meteors))
```

```
df['BLEU_Score'] = bleus
df['chrF_Score'] = chrfs
df['TER_Score'] = ters
df['METEOR_Score'] = meteors
df

"""# Check Pearson Correlation Coefficient r"""

# normalize bert_f1 and comet score?
# ...

score_a = df['Human_Score_A'].tolist()
score_b = df['Human_Score_B'].tolist()
avg_human_score = []
for i in range(len(score_a)):
    avg_human_score.append((score_a[i]+score_b[i])/2)
print(len(avg_human_score))
# average the scores into avg_human_score

# calculate pearson correlation -1 ~ 1 with new averaged DA scores
from scipy import stats
print("Pearson (Sentence BLEU): ", stats.pearsonr(avg_human_score, df['BLEU_Score'].tolist())[0])
print("Pearson (TER): ", stats.pearsonr(avg_human_score, df['TER_Score'].tolist())[0])
print("Pearson (METEOR): ", stats.pearsonr(avg_human_score, df['METEOR_Score'].tolist())[0])
print("Pearson (chrF): ", stats.pearsonr(avg_human_score, df['chrF_Score'].tolist())[0])
print("Pearson (BERTScore F1): ", stats.pearsonr(avg_human_score, df['BertScore_F1'].tolist())[0])
print("Pearson (BERTScore Precision): ", stats.pearsonr(avg_human_score,
df['BertScore_Prec'].tolist())[0])
print("Pearson (BERTScore Recall): ", stats.pearsonr(avg_human_score, df['BertScore_Recall'].tolist())[0])
print("Pearson (COMET): ", stats.pearsonr(avg_human_score, df['Comet_Score'].tolist())[0])

# pearson correlation between human score A and B
print("Pearson (DA Score A and B): ", stats.pearsonr(score_a, score_b)[0])
# what's a good correlation coefficient for human annotations?

"""# Evaluate three models with every metric
```

```
## Load translation results
"""

ours = pd.read_csv("results/bsd_dev_translations_our_model.tsv", sep='¥t')
m2m100 = pd.read_csv("results/bsd_dev_translations_m2m100.tsv", sep='¥t')
marianmt = pd.read_csv("results/bsd_dev_translations_marianmt.tsv", sep='¥t')

# Ours
ours_preds = ours['Translation_En'].tolist()
ours_refs = ours['Target_En'].tolist()
ours_srcs = ours['Source_Ja'].tolist()

ours_refs2d = []
for ref in ours_refs:
        ours_refs2d.append([ref])

st = time.time()
ours_bertscore = bertscore.compute(predictions=ours_preds, references=ours_refs, lang="en")
print('bertscore elapsed time: ', time.time()-st)
st = time.time()
ours_comet = comet_metric.compute(predictions=ours_preds, references=ours_refs, sources=ours_srcs)
print('comet elapsed time: ', time.time()-st)
st = time.time()
ours_bleu = bleu.compute(predictions=ours_preds, references=ours_refs2d)
print('bleu elapsed time: ', time.time()-st)
st = time.time()
ours_ter = ter.compute(predictions=ours_preds, references=ours_refs2d)
print('ter elapsed time: ', time.time()-st)
st = time.time()
ours_meteor = meteor.compute(predictions=ours_preds, references=ours_refs2d)
print('meteor elapsed time: ', time.time()-st)
st = time.time()
ours_chrf = chrf.compute(predictions=ours_preds, references=ours_refs2d)
print('chrf elapsed time: ', time.time()-st)
st = time.time()
```

```
print("¥n¥n¥n")

print('bertscore', ours_bertscore)
print('comet', ours_comet)
print('bleu', ours_bleu)
print('ter', ours_ter)
print('meteor', ours_meteor)
print('chrf', ours_chrf)

# m2m100
m2m100_preds = m2m100['Translation_En'].tolist()
m2m100_refs = m2m100['Target_En'].tolist()
m2m100_srcs = m2m100['Source_Ja'].tolist()

m2m100_refs2d = []
for ref in m2m100_refs:
        m2m100_refs2d.append([ref])

st = time.time()
m2m100_bertscore = bertscore.compute(predictions=m2m100_preds, references=m2m100_refs,
lang="en")
print('bertscore elapsed time: ', time.time()-st)
st = time.time()
m2m100_comet = comet_metric.compute(predictions=m2m100_preds, references=m2m100_refs,
sources=m2m100_srcs)
print('comet elapsed time: ', time.time()-st)
st = time.time()
m2m100_bleu = bleu.compute(predictions=m2m100_preds, references=m2m100_refs2d)
print('bleu elapsed time: ', time.time()-st)
st = time.time()
m2m100_ter = ter.compute(predictions=m2m100_preds, references=m2m100_refs2d)
print('ter elapsed time: ', time.time()-st)
st = time.time()
m2m100_meteor = meteor.compute(predictions=m2m100_preds, references=m2m100_refs2d)
print('meteor elapsed time: ', time.time()-st)
```

```
st = time.time()
m2m100_chrf = chrf.compute(predictions=m2m100_preds, references=m2m100_refs2d)
print('chrf elapsed time: ', time.time()-st)
st = time.time()

print("¥n¥n¥n")

print('bertscore', m2m100_bertscore)
print('comet', m2m100_comet)
print('bleu', m2m100_bleu)
print('ter', m2m100_ter)
print('meteor', m2m100_meteor)
print('chrf', m2m100_chrf)

# marianmt
marianmt_preds = marianmt['Translation_En'].tolist()
marianmt_refs = marianmt['Target_En'].tolist()
marianmt_srcs = marianmt['Source_Ja'].tolist()

marianmt_refs2d = []
for ref in marianmt_refs:
        marianmt_refs2d.append([ref])

st = time.time()
marianmt_bertscore = bertscore.compute(predictions=marianmt_preds, references=marianmt_refs,
lang="en")
print('bertscore elapsed time: ', time.time()-st)
st = time.time()
marianmt_comet = comet_metric.compute(predictions=marianmt_preds, references=marianmt_refs,
sources=marianmt_srcs)
print('comet elapsed time: ', time.time()-st)
st = time.time()
marianmt_bleu = bleu.compute(predictions=marianmt_preds, references=marianmt_refs2d)
print('bleu elapsed time: ', time.time()-st)
st = time.time()
marianmt_ter = ter.compute(predictions=marianmt_preds, references=marianmt_refs2d)
```

```
print('ter elapsed time: ', time.time()-st)
st = time.time()
marianmt_meteor = meteor.compute(predictions=marianmt_preds, references=marianmt_refs2d)
print('meteor elapsed time: ', time.time()-st)
st = time.time()
marianmt_chrf = chrf.compute(predictions=marianmt_preds, references=marianmt_refs2d)
print('chrf elapsed time: ', time.time()-st)
st = time.time()

print("¥n¥n¥n")

print('bertscore', marianmt_bertscore)
print('comet', marianmt_comet)
print('bleu', marianmt_bleu)
print('ter', marianmt_ter)
print('meteor', marianmt_meteor)
print('chrf', marianmt_chrf)

def avrg(lst):
    return sum(lst)/len(lst)

print('bertscore', avrg(ours_bertscore['f1']))
print('bertscore', avrg(m2m100_bertscore['f1']))
print('bertscore', avrg(marianmt_bertscore['f1']))

print('comet', ours_comet['mean_score'])
print('comet', m2m100_comet['mean_score'])
print('comet', marianmt_comet['mean_score'])

print('bleu', ours_bleu['bleu'])
print('bleu', m2m100_bleu['bleu'])
print('bleu', marianmt_bleu['bleu'])

print('ter', ours_ter['score'])
print('ter', m2m100_ter['score'])
print('ter', marianmt_ter['score'])
print('meteor', ours_meteor['meteor'])
print('meteor', m2m100_meteor['meteor'])
print('meteor', marianmt_meteor['meteor'])

print('chrf', ours_chrf['score'])
print('chrf', m2m100_chrf['score'])
print('chrf', marianmt_chrf['score'])
```