International Journal of Service and Knowledge Management
International Institute of Applied Informatics
2024, Vol. 8, No. 2, IJSKM847

Generation and Comparison of Distributed Representations of Words from Japanese Wiktionary with BERT Sentence-Embedding

Ryota Nishiura [*], Seiji Tsuchiya [†], Hirokazu Watabe [†]

Abstract

In this paper, we introduce a novel approach to generate distributed representations of words. Our approach makes use of structured contents of Wiktionary, in contrast to existing methods such as Word2Vec, which are learned from unstructured corpora of plain text. Our approach generates distributed representations of headwords in Wiktionary by obtaining sentence embeddings of the corresponding contents of the entries with a pre-trained BERT model. We demonstrate that our proposed method outperforms a Word2Vec model in a XABC test, potentially due to its ability to leverage both the quality of a pre-trained BERT model and expert-moderated knowledge from Wiktionary. We also conducted a comparative study on a variety of different BERT models to find the model conditions most suitable for our purpose.

Keywords: distributed r epresentation, word embedding, Wikipedia, Wiktionary, BERT, Word2Vec

1 Introduction

In the field of information processing, the vectorization of knowledge has been a top priority to this day. In recent years, RAG [1] has attracted industrial and academic attention, enabling language models to generate text using external knowledge with vector-based methods that convert knowledge into vectors and store them in databases. The idea of storing knowledge as vectors has been discussed for many years and has become universal in natural language processing tasks.

Word embedding is a method of enabling computers to understand natural language by representing individual words as distributed numerical vectors. In the history of representing individual words for computers, the main contribution of the distributed representations is that they can be used to calculate word similarities. Prior to this, there were concepts of representing words with integers and one-hot vectors, both of which can enable computers to identify individual words but not to calculate word similarities, as the distances of arbitrary integers by no means

[*] Graduate School of Science and Engineering, Doshisha University, Kyoto, Japan
[†] Doshisha University, Kyoto, Japan

indicate any similarity of words and all the distances of one-hot vectors are constant, assuming all the words are equally similar. Specifically, the dimension of one-hot vectors equals to the size of vocabularies, meaning that empirical representations with this method can have tens or hundreds of thousands of dimensions for each numerical vector, requiring significant computational resources. The contribution of distributed representations is that they solved these problems with reducing the number of the dimensions while encoding semantic patterns of words into the vectors, allowing for the calculation of word similarities.

The usage of word embedding includes a wide range of applications, including language translation, text classification, and sentiment analysis. The use of word embeddings has significantly improved the performance and the efficiency of these tasks, as they provide computers with better understandings on the semantic relationships among the words in natural languages. Additionally, they can help solve the problem of data sparsity, where there may be insufficient data available for a particular task. Therefore, improving and contributing to the realm of word embedding is crucial for advancing natural language processing research and developing more accurate and effective solutions.

Word2Vec [2] is a widely-used neural network-based method for word embedding, which is based on the algorithm to find co-occurrence patterns of words in a huge dataset of text corpus. There are the Continuous Bag of Words (CBOW) model and the Skip-gram model for Word2Vec implementations. The CBOW model is trained on the task of estimating the target words from surrounding continuous parts of text, while the Skip-gram model, inversely, is trained on the task of estimating the continuous parts of text from their central words. In both models, the distributed representation of a word is obtained from the hidden state of the neural network, which is a vector with empirically only hundreds of dimensions, which is much smaller than the size of the vocabulary of tens or hundreds of thousands. The semantic similarities of these vectors are derived from the co-occurrence patterns found between continuous sequences and words.

Despite their success in many natural language processing tasks, the co-occurrence patterns approach has a challenge regarding the fact that it relies on large corpora in unstructured format of text, leading to a problem with making partial changes to the text. This means that adding new words to an existing Word2Vec model is unrealistic in practice: 1) substantial usage of new words must be prepared in plain text, while the amount enough to maintain the quality of the representations are hard to estimate, and 2) models re-learned on additional corpora with new words change weights and biases in their networks thus the representation vectors of all the words, resulting in systems with re-learned models being not guaranteed to operate the same as the former ones. Therefore, the co-occurrence patterns approach has limitations in terms of its flexibility and compatibility when it comes to adapting to new data or incorporating new words into existing models.

In addition to the said limitations, Word2Vec are referred to as “bag of words” models, only considering the co-occurred words surrounding target words to train the weights and biases of neural networks, ignoring the order of these words. This means that the models do not take into account the sequential characteristics of words in sentences or documents, which includes word relationships such as syntax, grammar, and modification relationships, but instead treat each word from text as an independent entity. While this approach has been successful in many natural language processing tasks, it can have limitations in capturing complexities of language.

To address these limitations, there needs to be an approach which is robust to the change of corpora, allowing for adding or modifying representations of words without affecting existing ones, as well as being able to process not only co-occurrence patterns of independent words, but also complex relationships in sequential words of sentences.

This is why we have introduced a combined approach of pre-trained BERT [3] models and Wiktionary[1], a community-driven online dictionary hosted by Wikimedia Foundation. The main advantage of Wiktionary is the structure of headwords and their separate entries, and it is also easy to add and edit entries independently while maintaining their quality through expert moderation. BERT is a pre-trained transformer-based model for multiple natural language processing tasks and can be used to obtain text embeddings mainly for text classifications, which we have chosen to encode Wiktionary contents.

In this paper, we propose an approach to obtain distributed representations of Wiktionary headwords by encoding their entry contents with a pre-trained BERT model. Our approach offers greater flexibility and compatibility since individual words can be added or removed without the need for re-learning, while providing high capabilities for calculating word similarities, capturing complex patterns of language.

The remainder of this paper is composed as follows. The related work for our study is introduced in Chapter 2. As for the approach in our study, the methodology is described in Chapter 3, while the experiment is described in Chapter 4. The result and the discussion for the experiment are provided in Chapter 5 and 6. In Chapter 7, we conduct extended research with a variety of BERT implementations to obtain more general insights around our approach.

2 Related Work

2.1 Word2Vec

Word2vec [2] is a neural network-based algorithm for generating word embeddings, which are numerical representations of words in a high-dimensional vector space. The algorithm is used to learn distributed representations of words based on their co-occurrence patterns in large datasets of text. Word2vec has become popular because of its usefulness for a wide range of natural language processing tasks, such as language translation, text classification, and sentiment analysis.

There are two neural networks used in Word2Vec: the Continuous Bag of Words (CBOW) model and the Skip-gram model. Both models share a common approach relatable to autoencoder [4] models. The CBOW model is trained on the task of estimating the central word from the continuous parts of sentences. It takes the context words as input and tries to predict the central word. On the other hand, the Skip-gram model is trained on the task of estimating the continuous parts of sentences from the central word. It takes a central word as input and tries to predict the surrounding words. In both models, the distributed representation of words is obtained from the hidden state of the neural network, resulting in the dimensionality of hundreds of representational vectors of words. The vectors retain similarities between words because, as inferred from the architecture of autoencoder models, the lower-dimensional hidden state contains compressed

[1] https://www.wiktionary.org

understandings of the contextual co-occurrence patterns of texts.

As mentioned in the introduction, the Word2Vec learning approach with co-occurrence patterns has limitations when it comes to adapting to additional corpora or adding new words to existing models, as the change affects the weights and biases of the neural network models, making the representation of words incompatible with previous models. Thus, we focus on making our new approach to be robust on the partial change to text resources, which allows for adding or modifying individual words without affecting existing representations of words.

Furthermore, the Word2Vec algorithms only consider the sequence of specific lengths around the target words, disregarding the order of words. This means it may struggle to recognize the syntax, grammar, and modification relationships necessary to fully understand the complexity of language. While this algorithm has been successful in many natural language processing tasks, we aim to make our approach to leverage the complex relationships of words to make better representations of words, expanding the capability to tackle even more challenging natural language processing tasks.

To evaluate the effectiveness of our proposed approach, we conduct a comparative analysis between the Word2Vec model and our method of combining BERT and Wiktionary in terms of word similarity performance. Specifically, we measure the similarity scores for a set of words using both approaches and compare their results using cosine similarity. This evaluation allows us to determine the effectiveness and potential advantages of our proposed approach over the traditional Word2Vec model.

2.2 BERT

BERT (Bidirectional Encoder Representations from Transformers) [3] is a transformer-based neural network architecture that has revolutionized the field of natural language processing. Unlike other language models, BERT utilizes a bidirectional approach in training its model, which means it takes into account not only the preceding words, but also the following words, to understand the context of a particular word in a sentence. This helps in generating more accurate word embeddings and makes it possible to capture more complex linguistic relationships.

BERT was first introduced by Google in 2018 [3], and has since become one of the most popular and widely-used models in the field. It is pre-trained on a large corpus of text data, which allows it to learn general language patterns and relationships. This pre-training makes it possible to fine-tune the model on specific natural language processing tasks, such as question-answering, sentiment analysis, and language translation, by adding task-specific layers to the pre-trained model.

One of the key advantages of BERT is its ability to understand the nuances of natural language, including idiomatic expressions and the subtle meanings of words in different contexts. This has resulted in state-of-the-art performance on a variety of natural language understanding benchmarks, such as the GLUE benchmark and the SuperGLUE benchmark. As a result, BERT has become a popular choice for natural language processing tasks in both industry and academia.

As we aim to make computers understand natural language words from Wiktionary contents by leveraging not only the individual words but also relationships between words such as syntax, grammar, and modification relationships, BERT's ability to create high-quality text embeddings is suitable for our purpose. By utilizing BERT to encode the content of individual entries in

Wiktionary, we are able to obtain distributed representations for each headword that take into account not only the meaning of the headword itself, but also the contextual relationships between sentence words from the entry.

In our approach, we utilize a pre-trained BERT model for the Japanese language to encode Japanese Wiktionary entries. The model is trained on Japanese Wikipedia corpus and is capable of encoding natural language text into distributed representations, allowing us to obtain embeddings for Japanese words and phrases in Wiktionary.

3 Methodology

In our proposed method, we aim to overcome the limitations of traditional word embedding methods by combining the strengths of a pre-trained BERT model and the high-quality articles in Wiktionary. By using a pre-trained BERT model to encode the content of individual entries in Wiktionary, we can obtain distributed representations for each headword that capture more complex language patterns and relationships. This approach allows for a more nuanced understanding of words, leading to improved performance in natural language processing tasks.

Compared to the existing methods of distributed representations with the co-occurrence patterns approach like Word2Vec, our method offers several advantages as follows:

- The flexibility and compatibility, allowing for adding or modifying individual words without re-training entire models or affecting existing word representations. This is possible because those representations are derived independently from the corresponding Wiktionary entries, which also means it is possible to modify a specific Wiktionary entry and improve only the respective representation without affecting other word representations.

- The ability to leverage expert-moderated knowledge effectively and efficiently for computers to understand natural language words, by using Wiktionary, a well-curated online dictionary which provides a vast amount of expert-moderated knowledge, including definitions, synonyms, antonyms, and other relationships between words that were carefully curated by linguists and language experts, which may help computers to find important knowledge that is important but not necessarily easily learned from daily language usage.

- The utility of BERT as the encoding method, allowing for recognizing not only the meaning of target words but also complex but informative relationships between words such as syntax, grammar, and modification relationships from Wiktionary entries, which enhances the quality of the resulting headword embeddings.

The method consists of two processes. Firstly, we process a dump file of Wiktionary to create a set of pairs of headwords and their entries. Secondly, we input the set of pairs into the pre-trained BERT model to obtain encoded representations of the entries, which are then associated with their respective headwords. Fig. 1 shows the entire process for creating distributed representations of headwords with our proposed method. In the following sections, we will provide a detailed description of our methodology, including the preprocessing steps, model architecture, and evaluation metrics.

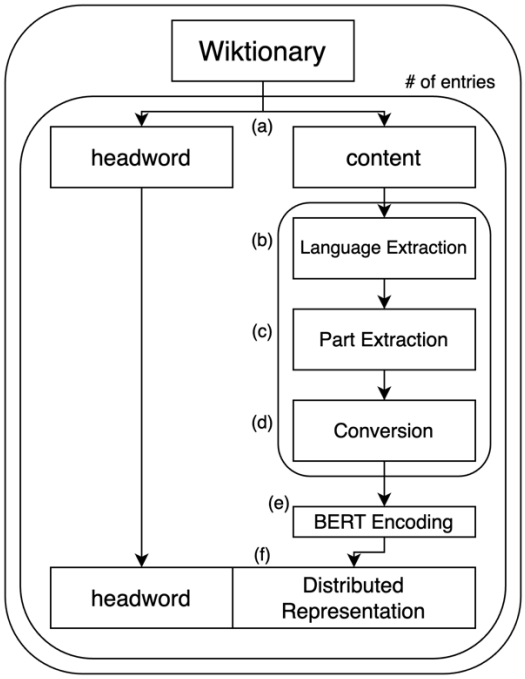

Fig. 1: The entire process for creating distributed representations of headwords with our proposed method.

3.1 Preprocessing Wiktionary Dump File

The preprocessing step is crucial for our method, as it allows us to take advantage of the structure and organization of Wiktionary's entries, making it possible to obtain high-quality embeddings of individual words.

We have used the 20230220 version of the Japanese Wiktionary dump file and processed it using BeautifulSoup4[2], a HTML/XML parser library for Python. Using the library, the headwords and their respective contents were extracted from the Wiktionary entries (Fig. 1, (a)).

The extracted contents were written in Wikicode, a markup language for MediaWiki contents. To further extract the necessary parts of the contents, we followed three subprocesses:

- **Targeted Language Extraction** (Fig. 1, (b)). Kanji idioms are used in several East Asian languages and Japanese Wiktionary has sections for each language. As we are focusing on the Japanese language to input in Japanese BERT models, we only extracted and used the Japanese sections from the contents.
- **Targeted Part Extraction** (Fig. 1, (c)). The Wiktionary entries include various information such as translations, pronunciations, examples, and etymologies, which are not necessary for calculating word similarities. Therefore, we extracted and used the meanings(definitions), synonyms, antonyms, idioms, relevant words, compound words, and driven words.

[2] https://www.crummy.com/software/BeautifulSoup

- **Converting Wikicode into Plain Text** (Fig. 1, (d)). In prior to inputting the contents to BERT models, they must be in non-marked up plain text, which is the same style as the dataset the BERT model was pre-trained on. To achieve this, we used mwparserfromhell[3] 0.6.4, a Wikicode parser library for Python, resulting in plain text as shown in Fig. 2.

Unprocessed

```
{{Wikipedia|感情}}
== {{ja}} ==
[[カテゴリ:{{ja}}|かんしよう かんじょう]]
{{ja-DEFAULTSORT|かんじょう}}
=== {{noun}} ===
[[カテゴリ:{{ja}}_{{noun}}|かんしよう かんじょう]]
'''[[感]] [[情]]''' （[[かんじょう]]）
#[[外界]]の[[刺激]]に[[応]]じ、一時的に[[変化]]する[[内心]]の[[状態]]。[[きもち|気持ち]]。[[理性]]や[[論理]]と異なり、[[本能]]的又は[[反射]]的に生ずる。
#[[冷静さ]]を失い[[怒り]]や[[悲しみ]]などにとらわれた[[状態]]。
#*'''感情'''に任せて怒鳴り散らす
#*一時の'''感情'''に突き動かされる

====={{pron|ja}}=====
;かˀんじょー
====={{rel}}=====
{{rel-top4}}
*[[悪感情]]
*[[感情移入]]
*[[感情線]]
*[[感情的]]
*[[感情に走る]]
*[[感情論]]
*[[国民感情]]
*[[処罰感情]]
*[[生活感情]]
*[[対立感情]]
*[[地域感情]]
*[[被害感情]]
*[[報復感情]]
*[[恋愛感情]]
{{rel-bottom}}
[[Category:{{ja}} 感情|*]]

=={{ko}}==
…
```

Processed

感情。外界の刺激に応じ、一時的に変化する内心の状態。気持ち。理性や論理と異なり、本能的又は反射的に生ずる。冷静さを失い怒りや悲しみなどにとらわれた状態。関連語・・・・悪感情 感情移入 感情線 感情的 感情に走る 感情論 国民感情 処罰感情 生活感情 対立感情 地域感情 被害感情 報復感情 恋愛感情 *。

Fig. 2: An example of how Wikicode of a Wiktionary entry is processed to the rendered content.

[3] https://github.com/earwig/mwparserfromhell

3.2 Generate Distributed Representations

We utilize the pre-trained Japanese BERT model to encode the processed contents to obtain representations of the headwords, we used the CLS token state, which is the first token of the final (12th) hidden layer. Previous studies have suggested that the CLS token values of the final (12th) hidden layer of a BERT model can be used for text embedding, as it captures overall meaning of input sentence or document.

In our proposed method, the processed texts corresponding to individual headwords are input into the BERT model (Fig. 1, (e)). We limited the token size of each text by 512 and any excess tokens were truncated, according to the configuration of the model. By extracting the CLS token state, we obtained 768-dimensional vectors as distributed representations for 51,608 headwords (Fig. 1, (f)).

4 Experiment

4.1 XABC Test

We conducted an evaluation of our proposed method using a XABC test [8]. This test is specifically designed to assess a computer's ability to accurately identify word similarities. The test comprises 1,780 sets of four words - X, A, B, and C. Words A, B, and C are arranged by humans in descending order of their similarity to word X, which represents the correct answer. This means, words X and A are highly relatable, while words X and B are moderately relatable and words X and C are not relatable in human sense.

During the evaluation, our method was tested on various sets of XABC words, and the number of problems in which our method arranged A, B, and C in the same order as the correct answer represented our method's score for the ability to recognize word similarities. In other words, a problem is answered correct if the cosine similarities of word X to words A, B, and C are descending in this order.

The specific XABC test consisted of 1,780 sets of problems, part of which are shown in Table 1, expressed in original Japanese words with alphabetical notations. For example, in the second case from the table, word X: 銀世界 (ginsekai) is a poetic expression for snow scenes. Word A: 雪景色 (yukigeshiki) also means snow scenes and is the word most relatable to word X: 銀世界 among the words A, B and C. Word B: 冬 (fuyu) means winter and is moderately relatable to word X, while word C: 塩水選 (ensuisen) is a seed sorting method using salt water and is least relatable to word X.

In this experiment, we compared the performance of our proposed method with that of a Word2Vec skip-gram model trained on a Japanese Wikipedia corpus using the gensim, a Python library for natural language processing and machine learning. The Word2Vec model was trained on a corpus of Japanese Wikipedia articles, the same condition as the BERT model employed in our method, which makes it a suitable benchmark for comparison with our method. We used WikiExtractor to extract plain texts from a dump file of the Wikipedia Corpus and the MeCab [5] parser to tokenize the texts. We set the dimensionality of the Word2Vec model to 200. As a result, we obtained 259,189 vectors of words. By comparing the results of our method with those of the

Table 1: Example of the XABC test set

		word	reading	meaning
1	X	勝ち	kachi	win
	A	勝利	shōri	victory / triumph
	B	試合	shiai	match / game
	C	延引	en'in	postponement
2	X	銀世界	ginsekai	silvery snow scene
	A	雪景色	yukigeshiki	snowy landscape
	B	冬	fuyu	winter
	C	塩水選	ensuisen	a seed selection method using saltwater
3	X	乗法	jōhō	multiplication (formal)
	A	掛け算	kakezan	multiplication (casual)
	B	算数	sansū	mathematics / arithmetic
	C	押し進める	oshi-susumeru	push ahead
4	X	雑穀	zakkoku	miscellaneous grain / cereals
	A	穀類	kokurui	grain / cereal
	B	農業	nōgyō	farming / agriculture
	C	横櫛	yokogushi	a comb worn crosswise for holding one's hair in place, the action of wearing a comb crosswise
5	X	絶体絶命	zettai-zetsumei	crisis from which one cannot escape
	A	危機	kiki	in danger/ crisis/ a critical moment
	B	逃れる	nogareru	escape / avoid / evade
	C	横線	yokosen	horizontal line / transverse line
⋮				

Word2Vec model, we can assess the effectiveness of our proposed method in identifying word similarities in the Japanese language.

4.2 Text Encoder Model

To encode text from Wiktionary, we utilized the pre-trained implementation of Japanese BERT model (cl-tohoku/bert-base-japanese-whole-word-masking) [6], which is available on Hugging Face[4] and trained on large text of Japanese Wikipedia Corpus. The Japanese BERT model has the same architecture as the original BERT base model [3]; 12 transformer layers of 12 attention heads with 768-dimensional hidden states. The pre-training data was Japanese Wikipedia as of September 1, 2019. WikiExtractor was used to extract plain texts from a dump file of Wikipedia articles, consisting of approximately 17M sentences. The texts are tokenized by MeCab morphological parser with the IPA dictionary [7] and then split into subwords by the WordPiece algorithm, resulting in the vocabulary size of 32,000. The model is trained with the same configuration as the original BERT; 512 tokens per instance, 256 instances per batch, and 1M training steps. For the training of the masked language modeling, the Whole Word Masking method was introduced where all the subword tokens corresponding to a single word are masked at once.

[4] https://huggingface.co

Table 2: The result of the experiment

Models	**Original**	**Inclusive**
Word2Vec	67% (331/499)	67% (56/84)
Our Method	**75%** (98/130)	**74%** (62/84)

5 Result

We conducted the experiment to evaluate the performance of our proposed method and compared it to a genism Word2Vec model trained on Japanese Wikipedia corpus. The result is presented in two parts: the Original test and the Inclusive test. The Original test included all the problems in the test set, while the Inclusive test included only the problems with their words that were present in both models.

The results of the XABC test are shown in Table 2, which includes the accuracy rate for the tests, as well as the numbers of problems solved and attempted, presented in the parentheses. The results showed that our proposed method achieved higher accuracy rates (number of correct answers per number of attempted problems) of 75% in the Original test and 74% in the Inclusive test than those of the Word2Vec model of 67% in both tests, demonstrating its high capability to identify word similarities.

6 Discussion

The results obtained from our experiments demonstrate that the proposed method for calculating word similarities in the XABC test is more effective than the Word2Vec model.

One possible reason that our proposed method showed better results may be attributed to the utilization a pre-trained BERT model, which is known for its ability to recognize not only separate words, but also complex relationships between words, including syntax, grammar, and modification relationships from Wiktionary entries. This allowed our method to generate more informative representations compared to the existing method, resulting in better word embeddings. However, it is worth noting that the current BERT model used in this study may not be the best available encoder, and there may be even more advanced models in the future that can generate even more informative representations, leading to even better results. Thus, further research in this area is necessary to explore the potential of using more advanced encoders for word similarity calculations.

Additionally, fine-tuning the BERT model for this specific task could potentially further improve the results. By fine-tuning the BERT model specifically for the task of word similarity calculation using Wiktionary data, we could potentially improve the performance of the proposed method by adjusting the model's weights to better fit the specific requirements of the task at hand. This process would involve feeding in Wiktionary data to the BERT model and adjusting its weights based on the training data to improve its performance on the word similarity calculation task. This fine-tuning process could potentially result in more accurate and informative representations of words, leading to further improvements in the accuracy of word similarity calculations. Overall, fine-tuning the BERT model is a promising approach for improving the proposed method and is worth exploring in future research.

Another possible reason that our proposed method showed better results may be that the quality of the content of Wiktionary had impact on the performance of the representations, as more accurate and comprehensive text would make a performance difference in the representations. Wiktionary contains detailed and accurate information, not easily apparent in daily usage, which the existing models like Word2Vec may struggle to capture. This is important as it can improve the performance of natural language processing tasks that require a deeper understanding of language.

It is important to note, however, that it is unlikely that the proposed method replaces the existing methods because the co-occurrence pattern approach have their own characteristics and are already used in many natural language processing tasks successfully. Additionally, the proposed method relies heavily on the human-curated contents of Wiktionary, which may not be able to cover all possible language patterns and relationships including the ones the co-occurrence pattern approach could make use of. However, the co-occurrence pattern approach has its own problem that it cannot update or improve their knowledge on demand: it is hard to predict the effect of updating training resources to a system and is impossible to directly add, delete, or update an exact portion of knowledge in need of change. These make it quite difficult to update running systems using this approach, diverging from an ideal state as languages change in month and years. Therefore, it may be better to consider combining or concatenating our method with the existing ones with the aim to take both advantages, rather than discussing whether it can completely replace them.

Despite the good performance in the result, the drawback of this proposed method is its limited size of vocabulary, which is bound to the size of Wiktionary itself. In comparison, the Word2Vec model had a vocabulary size account for 499 problems, while the proposed method only had a vocabulary size account for 130 problems, which appears anchored to the ratio of the vocabulary sizes of both models, 259,189 words and 51,608 words, respectively. It can turn out to be problematic on practical natural language processing tasks. Meanwhile, a comparison between the Original and Inclusive tests suggests that vocabulary size does not significantly impact accuracy rate, meaning that it may be possible that even as Wiktionary continues to grow, the performance of the proposed method may remain the same, higher than existing methods. However, it is possible that the proposed method may face challenges in scaling up to larger vocabularies, as Wiktionary contents being maintained by human resources thus their growth rate is limited. As a result, scaling up the method to larger vocabularies may prove to be difficult using only Wiktionary as the source of structured data.

To overcome this limitation, it may be necessary to incorporate other online resources that provide structured data, to supplement Wiktionary's content. This could potentially increase the amount of available data and improve the quality of the word embeddings generated by the method. However, this would also require careful consideration of data quality and consistency across different sources, as well as potential challenges in merging and processing data from multiple sources. Therefore, further research is needed to investigate the feasibility and effectiveness of combining multiple resources to scale up the proposed method.

It is also important to note that the conclusion is limited to the specific test which was used in this study and may not necessarily hold true for all scenarios. The performance of the proposed method could be affected by various factors, such as the steps and quality of the processing data, the language used for evaluation, and the specific natural language processing task at hand. There needs to be further research to better understand this concern and draw more general conclusions.

7 Comparative Study

Since there are a variety of different pre-trained Japanese BERT implementations available, we have conducted the additional experiment on these models to find which models are more suitable for our proposed method and analyze what makes difference to the performances.

In the following are shown the 12 pre-trained implementations of Japanese BERT models with different conditions compared in our additional experiment:

(1) **bert-base-japanese:** the original BERT model from the cl-tohoku repository, trained on Japanese Wikipedia as of September 1, 2019. The input texts are first tokenized by MeCab morphological parser with the IPA dictionary and then split into subwords by the WordPiece algorithm. The vocabulary size is 32,000.

(2) **bert-base-japanese-whole-word-masking:** retaining conditions from the **(1) bert-base-japanese** model as the foundation, the whole word masking method was introduced where all the subword tokens corresponding to a single word are masked at once. This model is the one also used in the former experiment in our study.

(3) **bert-base-japanese-char:** retaining conditions from the original **(1) bert-base-japanese** model as the foundation, the character-level tokenization is used in this version: the texts are first tokenized by MeCab morphological parser with the IPA dictionary and then split into characters. The vocabulary size is fewer 4,000, as a result of using the character-level tokenization.

(4) **bert-base-japanese-char-whole-word-masking:** retaining conditions from the **(1) bert-base-japanese** model as the foundation, the combination of the character-level tokenization and the whole word masking method are introduced. The vocabulary size is 4,000.

(5) **bert-base-japanese-v2:** retaining conditions from the **(2) bert-base-japanese-whole-word-masking** model as the foundation, the training corpus and the tokenization method are changed in this version. The training corpus of the Japanese version of Wikipedia is generated from the Wikipedia Cirrussearch dump file as of August 31, 2020. The texts are first tokenized by MeCab with the Unidic 2.1.2 dictionary and then split into subwords by the WordPiece algorithm. The vocabulary size is 32,768.

(6) **bert-base-japanese-char-v2:** retaining conditions from the **(5) bert-base-japanese-v2** model as the foundation, the character-level tokenization is used in this version, resulting in the vocabulary size of 4,000.

(7) **bert-base-japanese-v3:** retaining conditions from the **(5) bert-base-japanese-v2** model as the foundation, the training corpus is changed in this version. Trained on the Japanese portion of CC-100 dataset and the Japanese version of Wikipedia. For Wikipedia, a text corpus is generated from the Wikipedia Cirrussearch dump file as of January 2, 2023. The vocabulary size is 32,768, the same as the foundation model.

(8) **bert-large-japanese-char-v3**: retaining conditions from the **(7) bert-base-japanese-v3** model as the foundation, the character-level tokenization is used in this version. The vocabulary size is 4,000.

(9) bert-large-japanese: retaining conditions from the **(5) bert-base-japanese-v2** model as the foundation, the model architecture is the same as the BERT large model; 24 layers, 1,024 dimensions of hidden states, and 16 attention heads.

(10) bert-large-japanese-char: retaining conditions from the **(9) bert-large-japanese** model as the foundation, the character-level tokenization is used in this version. The vocabulary size is 4,000.

(11) bert-large-japanese-v2: retaining conditions from the **(7) bert-base-japanese-v3** model as the foundation, the model architecture is the same as the BERT large model; 24 layers, 1,024 dimensions of hidden states, and 16 attention heads.

(12) bert-large-japanese-char-v2: retaining conditions from the **(11) bert-large-japanese-v2** model as the foundation, the character-level tokenization is applied in this version. The vocabulary size is 4,000.

In summary, these models vary in terms of the sizes of the model architectures, the tokenization methods, the use of character-level tokenization along with the vocabulary size, the use of whole word masking method, and the generations of training corpus. Our purpose of the additional study is to find the most suitable conditions from them for generating the represented distributions of headwords with our approach.

The result of the extended experiment of our approach is shown in Fig. 3. The numbers of attempted problems by each model were 130 in the Original test and 84 in the Inclusive test, out of 1,780 sets of the XABC test, the same as the numbers shown in our former experiment as the vocabulary size of our method is bound to the number of entries in Wiktionary, regardless

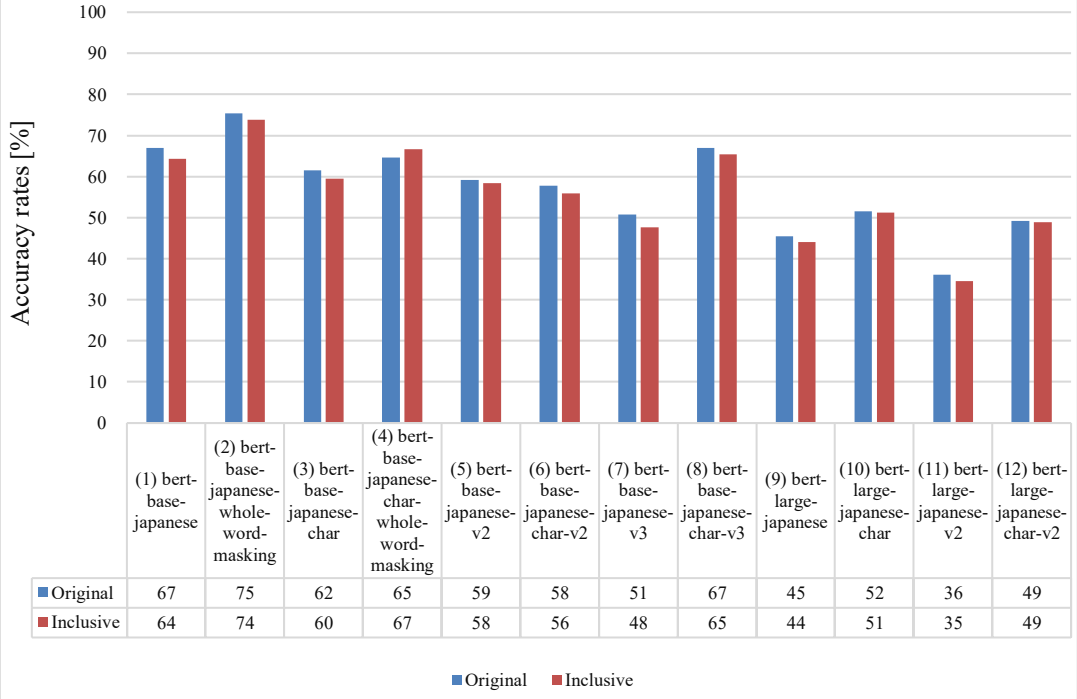

Fig. 3: The accuracy scores for XABC tests with different BERT models used for encoding.

of the difference of the encoder models.

Fig. 3 shows the model achieving the best performance on the XABC test was **(2) bert-base-japanese-whole-word-masking**, which is also used in the former experiment to get compared with a Word2Vec model, with accuracy of 75% and 74% in each test. Meanwhile, the model with the worst performance was **(11) bert-large-japanese-v2,** with accuracy of 36% and 35%, less than half of the best ones.

The result shows that the **(2) bert-base-japanese-whole-word-masking** model is the most suitable for our approach. In the following discussion, we will analyze the reason for this outcome.

One possible reason that the model showed the best performance is the use of the whole-word-masking method. While the comparison is impossible in the newer generations of models as the method has become a prerequisite for them, the first generation **(2) bert-base-japanese-whole-word-masking** and **(4) bert-base-japanese-char-whole-word-masking,** with the whole word masking method applied, have outperformed **(1) bert-base-japanese** and **(3) bert-base-japanese-char** respectively**,** confirming the effectiveness of the method.

Another possible reason is the difference on the generations of the models with different training corpora and the tokenization methods. In the newer generations **(5)** - **(12)**, with the exception of **(8) bert-large-japanese-char-v3**, have underperformed the first generations **(1)** - **(4)**. Since the newer generations are served with larger and newer dataset from Wikipedia, the third generations **(7)**, **(8)**, **(11)**, and **(12)** being trained on the Japanese part of CC-100 dataset on top of that, the result confirms that models with updated datasets don't necessarily gain better performance. In other words, updating datasets for the purpose of improving the model is difficult to achieve with certainty. This discussion suggests the necessity of technologies where updating data affects the existing performance with minimal directed influences.

While it cannot be confirmed that the factor that declines the performance on the newer generations of models is the updated training data, as the interaction between the data and the initial weights and biases of the neural networks may have affected the inference patterns of models, the major tendency between these generations with a variety of models in the result supports the explanation that it is likely there are the causes for it in the updated training data itself.

The interaction of between the use of the character tokenization and the architectures with different scales, the base and large models, may be yet another factor for the result. With **(10) bert-large-japanese-char** and **(12) bert-large-japanese-char-v2**, both using the character tokenization, outperforming **(9) bert-large-japanese** and **(11) bert-large-japanese-v2** respectively, it is suggested that the large architecture with larger word embedding size and more of attention layers is too large for the scale of the specific training situation, resulting in these character tokenization models with the smaller vocabulary size of 4,000, having less parameters to train and information to contain, were more successful compared to the normal models with that of 32768. As for the base models, on the contrary, it is possible that the scale of their networks was small enough to leverage the merit of treating the Japanese language with the meaningful subword tokens instead of the character tokens with the scale of the training situation, over the simplicity of smaller vocabulary size, thus the models without the character tokenization outperforming their corresponding competitors, with the exception of **(8) bert-base-japanese-char-v3**. That is describing the reason the base models **(1)** – **(8)** performed better than the large models **(9)** – **(12)**.

While we could give the limited reason **(8) bert-base-japanese-char-v3** does not follow the tendency discussed here, it may be possible that the interaction between the scale of models and the use of the character tokenization is rather subtle in the base models when compared to the great effects discussed in the large models; it cannot be determined whether the character tokenization is of no use for base models in any case.

With the discussion above, it is concluded that **(2) bert-base-japanese-whole-word-masking** was the best performing model because it uses the whole-word-masking method, is trained on the first generation of training corpora and tokenization methods, and is a base model without using the character tokenization.

8 Conclusion

In this paper, we presented a novel approach for generating distributed representations of words. Unlike existing methods such as Word2Vec, which rely on unstructured datasets of plain text, our approach leverages the structured contents of Wiktionary. Specifically, we employed a pre-trained BERT model to obtain representation vectors of individual Wiktionary headwords by embedding the corresponding content of Wiktionary entries. Through our experiments, we demonstrated that our proposed method outperformed a Word2Vec model in an XABC test. The result may be attributed to our method's ability to leverage the quality of both a pre-trained BERT model and expert-moderated Wiktionary. It may be used to improve the performance of natural language processing tasks that require a deeper understanding of language. We also conducted comparative study on a variety of different BERT models to find the model conditions most suitable for our purpose.

We also acknowledge that our method has some limitations. One potential drawback is its limited vocabulary size, which is bound to the size of Wiktionary itself. Additionally, we need to evaluate the method on more tasks to fully understand its capabilities and limitations. Therefore, further research is required to draw more general conclusions about the effectiveness of our proposed method. We believe that this research has the potential to inspire new directions in the development of methods for distributed word representations, and we plan to continue exploring this topic in future work. By expanding our research in these areas, we hope to demonstrate the practical utility of this method and contribute to the advancement of the field of artificial intelligence.

References

[1] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474, 2020.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv preprint, arXiv:1301.3781, 2013.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint, arXiv:1810.04805, 2018.

[4] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” Science, vol. 313, no. 5786, pp. 504–507, 2006.

[5] T. Kudo, “MeCab: Yet another part-of-speech and morphological analyzer,” https://taku910.github.io/mecab/, Accessed Nov 20, 2024.

[6] M. Suzuki and R. Takahashi, “Pretrained Japanese BERT models,” https://www.nlp.ecei.tohoku.ac.jp/research/open-resources/, Accessed Nov 20, 2024.

[7] M. Asahara and Y. Matsumoto, “ipadic version 2.7.0 User’s Manual,” https://osdn.net/projects/naist-jdic/docs/ipadic-2.7.0-manual-en.pdf/en/1/ipadic-2.7.0-manual-en.pdf, Accessed Nov 20, 2024.

[8] N. Okumura, S. Tsuchiya, H. Watabe, and T. Kawaoka, “A construction of large-scale concept-base for calculation of degree of association between concepts,” The Association for Natural Language Processing, vol. 14, no. 5, pp. 41–64, 2007.