Improving Neural Language Processing with Named Entities

Kyoumoto Matsushita
Fujitsu Ltd. / Tokyo, Japan
m.kyoumoto@fujitsu.com

Takuya Makino
Fujitsu Ltd. / Tokyo, Japan
makino.takuya@fujitsu.com

Tomoya Iwakura
Fujitsu Ltd. / Tokyo, Japan
iwakura.tomoya@fujitsu.com

Abstract

Pretraining-based neural network models have demonstrated state-of-the-art (SOTA) performances on natural language processing (NLP) tasks. The most frequently used sentence representation for neural-based NLP methods is a sequence of subwords that is different from the sentence representation of non-neural methods that are created using basic NLP technologies, such as part-of-speech (POS) tagging, named entity (NE) recognition, and parsing. Most neural-based NLP models receive only vectors encoded from a sequence of subwords obtained from an input text. However, basic NLP information, such as POS tags, NEs, parsing results, etc, cannot be obtained explicitly from only the large unlabeled text used in pretraining-based models. This paper explores use of NEs on two Japanese tasks; document classification and headline generation using Transformer-based models, to reveal the effectiveness of basic NLP information. The experimental results with eight basic NEs and approximately 200 extended NEs show that NEs improve accuracy although a large pretraining-based model trained using 70 GB text data was used.

1 Introduction

In statistical NLP technologies, which were widely employed before neural-based NLP emerged, basic information, such as POS tags and NEs were often used as features for document classification (Higashinaka et al., 2012) and other NLP tasks. However, since neural-based NLP technologies have emerged with state-of-the-art (SOTA) performances, basic NLP technologies, such as POS tagging and NE recognition are no longer used for obtaining features from given text. This is because most neural-based NLP methods attain higher accuracy using only sentence representation encoded by a pretraining model learned from the large scale unlabeled text.

However, we think there are some rooms that basic information, such as POS tags and NEs contribute to recent neural-based NLP even if large scale pretrained models such as BERT (Devlin et al., 2019) and BART (Lewis et al., 2020) are used for obtaining sentence representations.

One of the reasons is that POS tags and NEs are usually obtained from outputs of analyzers trained from labeled training data created by human-beings for each specific purpose. For example, NE recognizers can identify single words or phrases with their classes, such as PERSON, ORGANIZATION, and so on, which are not explicitly given from pretrained models. Therefore, we think different kinds of information compared with pretraining ones are obtained from outputs of such NLP tools and we expect such information contributes to further improve accuracy.

We propose the incorporation of basic NLP information to neural NLP architectures and evaluate their effectiveness on two Japanese NLP tasks. In this paper, two NE categories, eight basic NEs (Sekine and Isahara, 2000) and approximately 200 extended NEs (Sekine and Nobata, 2004), are considered. By combining NE information with a pretrained model, we train a model of each task.

Experimental results on document classification using the BERT model and headline generation using the BART model, show that combining NEs with a large pretrained model contributes to significantly improved accuracy.

2 Experimental Design

This paper investigates the effectiveness of NE information for SOTA NLP models. In order to investigate effectiveness of NEs for SOTA NLP technologies, we conducted experiments on the two tasks. The first one is document classification for investigating the effectiveness on a classifcation task. The one is headline generation for investigating the effectiveness on a generation task. We use BERT (Devlin et al., 2019) for document classification and BART (Lewis et al., 2020) for headline generation as pretraining models.

To evaluate these tasks, we defined an architecture that uses NE class embeddings in addtion to subword embeddings given by one of the pretrained models, for each task. We refer to the architectures as BERT$^{\mathrm{NE}}$ and BART$^{\mathrm{NE}}$, where NE indicates the NE class definition. For the inputs of BERT$^{\mathrm{NE}}$ and BART$^{\mathrm{NE}}$, we recognize NEs in texts with an NE recognizer. Finally, using BERT$^{\mathrm{NE}}$ and BART$^{\mathrm{NE}}$, we evaluated the models with and without NE annotations on the two tasks for examining the effectiveness of NE annotations in each task.

One expected effect is the impact of granularity of NE classes on the accuracy of each task. To investigate such an effect, we use the following Japanese NE categories, described in Section 5.

- Basic Named Entity (BNE): eight types the basic NEs defined by the IREX committee (Sekine and Isahara, 2000).
- Extended Named Entity (ENE): approximately 200 types of ENE classes (Sekine and Nobata, 2004).

Furthermore, to evaluate the impact of NE recognition accuracy, we use the following two NE recognition methods.

- FNER: a feature-based NE recognizer (NER) (Iwakura, 2011)
- NNER: a neural-based NER (Akbik et al., 2018)

For FNER and NNER, we trained models of the two NE class definitions for each NE recognizer. Four NE recognizers were used in this experiment.

The accuracy of NNER exceeds that of FNER. In our internal evaluation with the IREX GENERAL data for the BNE definition, the accuracy of NNER for BNE is 93.44, which is 2.07 points higher F-measure than 91.37 of FNER. With these two NE recognizes, we investigate the impacts of NER accuracy on the performance of document classification and headline generation.

We use the following terms for the different settings.

$$\texttt{PRETRAIN}^{NEC}_{NER},$$

where PRETRAIN is BERT or BART, NEC is one of BNE and ENE, and NER is one of FNER and NNER. For example, BERT$^{ENE}_{FNER}$ indicates BERT with ENE classes gives from FNER.

3 Baseline Models

Here, we use two pretraining architectures, a pretrained BERT model for document classification and a pretrained BART model for headline generation.

3.1 BERT-based Document Classification

BERT is a Transformer-based pretraining architecture showing high performances in various NLP tasks. It generates encoded embeddings of each input token using a Transformer-based bidirectional encoder. Then, a target task architecture uses the embeddings.

For the document classification task, document classifiers use $\mathbf{h}_{[\mathrm{CLS}]}$, encoded embedding of [CLS] token by BERT, and classify the document based on $\mathbf{h}_{[\mathrm{CLS}]}$. This is because $\mathbf{h}_{[\mathrm{CLS}]}$ is used as the aggregate sequence representation for document classification task in BERT architecture.

$$\mathbf{o}_{[\mathrm{CLS}]} = \mathrm{classifier}(\mathbf{h}_{[\mathrm{CLS}]}),$$

where [CLS] is a special token corresponding to the beginning of input tokens; $\mathbf{h}_{[\mathrm{CLS}]} \in \mathbb{R}^D$ is an embedding of [CLS] token encoded with BERT; D is the number of dimensions of the encoded embeddings, $\mathbf{o}_{[\mathrm{CLS}]} \in \mathbb{R}^C$ is a score vector of classes and C is a number of document classes. We employ $\mathrm{argmax}(\mathbf{o}_{[\mathrm{CLS}]})$ as a predicted document class.

The embeddings of the [CLS] of each document is obtained with up to first 510 tokens in an input and [SEP] token that is also one of special tokens indicating the end of an input. The embeddings of each token of BERT is represented as

$$\mathbf{e}_{\mathrm{input}} = \mathbf{e}_{\mathrm{pos}} + \mathbf{e}_{\mathrm{ttype}} + \mathbf{e}_{\mathrm{token}}, \qquad (1)$$

where $\mathbf{e}_{\mathrm{pos}}$ is relative position embeddings for relative position pos of a token from the [CLS]; $\mathbf{e}_{\mathrm{ttype}}$ is token type embeddings for ttype indicating id (such as a natural number) of each sentence included in an input; $\mathbf{e}_{\mathrm{token}}$ is token embeddings for a token token. Figure 1 shows an example of generating input embeddings from input tokens.

3.2 BART-based Headline Generation

BART is also a Transformer-based pretraining architecture, especially for sequence-to-sequence

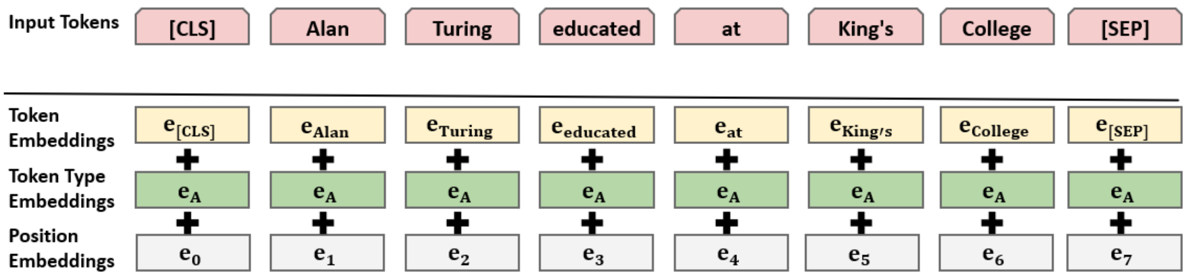

Figure 1: Example of generation of input embeddings of BERT.

tasks. This architecture shows high performance sequence-to-sequence tasks, such as headline generation and document summarization. BART composed of an encoder and a decoder in contrast to BERT, which consists of only an encoder. The pretraining follows an autoencoder approach in which an input sentence is generated from the input sentence. For example, “A B C” is an input, BART uses a variant of the sentence, in which words are shuffled like “B A C” or a masked sentence like “A _ C” as inputs for generating “A B C”.

BART encoder generates encoded contextual embeddings of subwords, $\mathbf{H}_{\text{enc}} \in \mathbb{R}^{D \times N}$, from an input subword sequence with a bidirectional encoder as in BERT, where D is a dimension of contextual embeddings and N is the length of an input sentence. Eq. (1) is used of each subword for the BART encoder.

Then, the BART decoder generates the i-th token t_i from the encoded embeddings and previously generated $i-1$ token embeddings with a left-to-right Transformer-based decoder,

$$t_i = \text{BART}_{\text{DEC}}(\mathbf{t}_{\text{dec},i-1}, \mathbf{H}_{\text{enc}}),$$

where t_i is the i-th token of the decoded tokens; $\mathbf{t}_{\text{dec},i}$ is i decoded tokens (i.e., $\mathbf{t}_{\text{dec},i} = (t_0, t_1, ..., t_{i-1})$). The t_0 is a special token representing the head of the decoded tokens and is denoted by “<s>”.

4 Named Entity-based Models

This section describes a BERT-based document classification and BART-based headline generation using NE information.

4.1 Use of NE Information

The difference for both base models is the representation of $\mathbf{e}_{\text{input}}$ defined by Eq. (1).

To incorporate NE information into BERT and the encoder of BART, we use $\mathbf{e}^{NE}_{\text{input}}$ instead of $\mathbf{e}_{\text{input}}$ defined as

$$\mathbf{e}^{NE}_{\text{input}} = \mathbf{e}_{\text{pos}} + \mathbf{e}_{\text{NE}} + \mathbf{e}_{\text{token}},$$

where $\mathbf{e}_{\text{NE}}$ corresponding to each NE class is use, instead of $\mathbf{e}_{\text{ttype}}$.

Figure 2 shows an example of input embeddings from input tokens with NE classes.

4.2 Parameter Updates for NE-based Models

Since we used $\mathbf{e}_{\text{NE}}$ instead of $\mathbf{e}_{\text{ttype}}$, accuracy may be degraded by using the same fixed pretrained parameters of BERT/BART. To avoid this degradation, we finetune all parameters of BERT/BART using the training data of the target task .[1]

Another option is training BERT and BART models with NE classes from scratch. However, we chose to finetune the original pretrained models that do not have $\mathbf{e}_{\text{NE}}$ for the following reasons:

- Our method takes advantage of existing pretraining models to easily incorporate different definitions of NEs since it does not require any pretraining to incorporate NEs.
- The maximum NE class types are approximately 200 of the ENE definition. Therefore, the finetuning approach can adopt NE embeddings based inputs.
- For the BERT, the use of a publicly available model is a fair comparison because the model was trained not for our purpose. [2]

The proposed method can be applied to other huge pretraining methods, such as RoBERTa (Liu et al., 2019) for document classification and PEGASUS (Zhang et al., 2020) for headline generation. Even if we cannot obtain the same huge pretraining dataset, we can use existing pretraining models to enhance them with NEs.

[1]The finetuning of BERT$^{\text{NE}}$ and BART$^{\text{NE}}$ from pretrained BERT and BART excludes $\mathbf{e}_{\text{NE}}$.

[2]Unfortunately, we could not find any models of BART for Japanese. Therefore, we trained a Japanese model for BART.

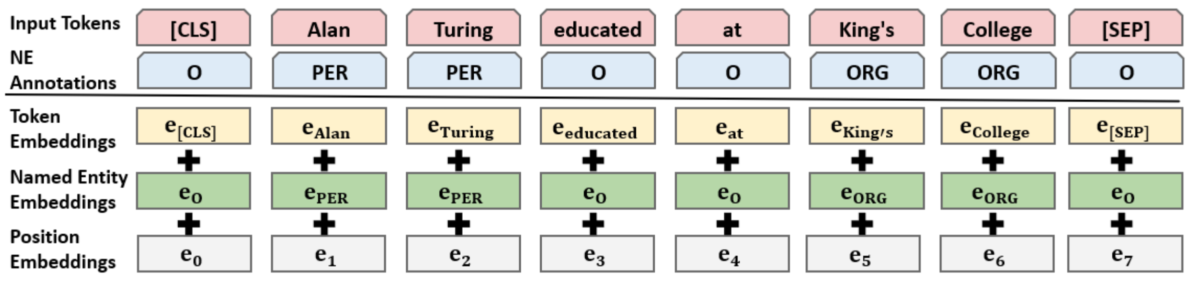

Figure 2: Example of generating input embeddings with NE information.

NE class	Example
ARTIFACT	Nobel Prize in Chemistry
LOCATION	Bulgaria
ORGANIZATION	King's College
PERSON	John Smith
DATE	May
MONEY	100 USD
PERCENT	100%
TIME	10:00 a.m.

Table 1: NE examples of BNE.

5 Named Entity Definition

This section introduces the two NE definitions used in our experiments, a basic NE category defined at Retrieval and Extraction Exercise (IREX) (Sekine and Isahara, 2000) and Extended Named Entity (ENE) definition (Sekine and Nobata, 2004).

- **Basic Named Entity (BNE)**: Table 1 shows an example of the BNE definition. BNE consists of eight NE classes, PERSON, ORGANIZATION, DATE, TIME, LOCATION, PERCENT, MONEY and ARTIFACT, and a special class OPTIONAL. The OPTIONAL is used when annotators cannot uniquely decide the NE class of each NE. In our experiments, we excluded the OPTIONAL NE annotations in the training and evaluation phases.

- **Extended Named Entity (ENE)**: The ENE definition has over 200 NE classes associated with a hierarchy. The IGNORE class of ENE represents the excluded parts. The CONCEPT class of ENE represents entities that cannot be classified into other ENE classes. In our experiments, we excluded IGNORE and CONCEPT NE classes in the training and evaluation phases.

Table 2 shows examples of NE class annotations with the two definitions. The ENE definition is more elaborated than that of BNE. For example, the BNE class of "King's College" is ORGANIZATION, however, that of ENE falls under School, which is a subcategory of ORGANIZATION.

6 Data Sets

This section introduces data sets for document classification and headline generation, created from the articles of Mainichi newspaper[3].

6.1 Document Classification Data Set

The document classification dataset was created from Mainichi newspaper data[4] (`Mai-news`). In this dataset, we used 2019 year articles as test data, 2018 year articles as development data, and articles of 2013-2017 years as training data. We used this split setting because, in practical situations, we usually have to train a model to predict future target information using past data. We used a pre-processed main body text of each `Mai-news` article by a procedure described in the Appendix, as an input text on document classification.

Table 3 shows the document classes. We used 14 categories of news articles as the target classes of our document classification task[5]. The document class of each article is recorded in its AD attribute. We refer to this dataset for document classification as `Mai-news-dc`. The second column of Table 4 shows the statistics of `Mai-news-dc`.

6.2 Headline Generation Data Set

The dataset for headline generation was also created from `Mai-news`. As in the document classi-

[3]This new paper is released from The Mainichi Newspapers Co., Ltd.

[4]https://www.nichigai.co.jp/sales/mainichi/mainichi-data.html

[5]We excluded four categories such as "front page", which are listed in Appendix.

NE category	Example of Annotations
Without NE	Alan Turing educated at King's College.
BNE	<PERSON>Alan Turing</PERSON> educated at <ORG>King's College</ORG>.
ENE	<Person>Alan Turing</Person> educated at <School>King's College</School>.

Table 2: Examples of annotation each NE category. "ORG" indicates ORGANIZATION NE class of BNE.

Document Class		
Commentary	Editorial	World
Economy	Special Topic	Culture
Household	Sports	Society
Science	Life	Entertainment
Multi Discipline	Reading books	

Table 3: Document classes used in document classification.

fication, we used pre-processed main body text of `Mai-news` as input text. We used the headline of each article as targets (i.e., headline models should generate). In `Mai-news`, the headline is identified by T1 attribute in each data corresponding to an article. We refer to this dataset for headline generation as `Mai-news-hg`. The third column of Table 4 shows the statistics of `Mai-news-hg`.

	`Mai-news`	
	DC	**HG**
train	378,779	426,214
dev	68,287	77,535
test	58,628	66,866

Table 4: The statistics of each dataset. "DC" and "HG" represent `Mai-news-dc` and `Mai-news-hg`, respectively.

7 Experimental Setting

This section shows the preprocess of input texts, hyperparameters, evaluation metrics, and so on.

7.1 Pretrained Models

We used cl-tohoku/bert-base-japanese-whole-word-masking[6] as our Japanese BERT pretrained model. This model was pretrained with approximately 17 million sentences of Japanese Wikipedia articles. The configuration is the same as the original BERT.

For BART, we used the 70 GB Japanese dataset of CC-100 [7] for pretraining BART. This is because no Japanese BART models were publicly available. The Japanese version of BART is pretrained 178,000 steps with a mini-batch size of 1,024 on 64 NVIDIA Volta 100 GPUs. For tokenizing texts, we used a unigram language model-based subword tokenization (Kudo and Richardson, 2018) trained on the same data used to pretrain the Japanese version of BART.

7.2 Training Models

We fine-tuned BERT$^{\mathrm{NE}}$ models using `Mai-news-dc` annotated by BNE and ENE categories, and BART$^{\mathrm{NE}}$ models with `Mai-news-hg` annotated by the two NE categories only one time. For each evaluation, these models were run once. The Appendix includes other hyperparameters.

7.3 Named Entity Annotation to Text

We annotated texts of `Mai-news-dc` and `Mai-news-hg` datasets with NE classes using NE recognizers, a classic feature-based NE recognizer (Iwakura, 2011) and a pretraining-based NE recognizer (Akbik et al., 2018). We refer to the former as FNER and the latter as NNER.

Each NE recognizer has two models; the BNE and ENE. The BNE-based models were trained using the IREX CRL dataset (Sekine and Isahara, 2000) and training data annotated by the authors. Also, the ENE-based models were trained using the GSK ENE training data [8].

7.3.1 Document Classification

First, tokenization of the input texts was performed. Here, a text is first tokenized by MeCab[9], a Japanese morphological analyzer that segments words with their POS tags from a text, with IPA dictionary[10]. Then, cl-tohoku/bert-base-japanese-whole-word-masking's tokenizer based on the WordPiece (Schuster and Nakajima, 2012) was applied to tokenized text with MeCab to decompose each word into subwords. We en-

[6]`https://huggingface.co/cl-tohoku/bert-base-japanese-whole-word-masking`

[7]http://data.statmt.org/cc-100/

[8]https://www.gsk.or.jp/catalog/gsk2014-a/

[9]`https://taku910.github.io/mecab/`

[10]`https://drive.google.com/uc?export=download&id=0B4y35FiV1wh7MWVlSDBCSXZMTXM`

hanced the cl-tohoku/bert-base-japanese-whole-word-masking's tokenizer with special tokens for NE classes such as "<PERSON>" and "</PERSON>". Then, each $\mathbf{e}_{\mathrm{NE}}$ was assigned to its corresponding subword and NE class tokens were removed.

After preprocessing, we fine-tuned a model not only targeting model-specific layers but also BERT layers.[11]

The model was evaluated using development data every 1,000 batch steps in a training phase and if the model achieves the best accuracy on the development data, we kept the model. The final accuracy of the experiments was calculated using the kept model.

7.3.2 Headline Generation

We tokenized the input source and target texts using the SentencePiece tokenizer described in section 7.1 and cut the texts by border characters (i.e., "<" and ">") between an NE class token, such as "<PERSON>" and other strings. Then, each $\mathbf{e}_{\mathrm{NE}}$ was assigned to its corresponding subword and NE class tokens were removed.

After preprocessing, we fine-tuned a model using BART$^{\mathrm{NE}}$ architecture.[12]

7.4 Evaluation Metrics

The following metrics were employed for evaluation.

Document Classification: We evaluated the outputs of document classification models using the macro F-measure of all classes.

Headline Generation: The outputs of headline generation models were evaluated using the F-measure of ROUGE-1 (R-1), ROUGE-2 (R-2), and ROUGE-L (R-L) (Lin, 2004), widely used as automatic evaluation metrics for headline generation. R-1 and R-2 are calculated based on overlap of uni-grams and bi-grams between a generated summary and its reference summary, respectively. Similarly, R-L is calculated based on overlap of the longest common subsequences between them.

8 Experimental Results

Table 5 shows the experimental results of document classification. Three models trained using NEs achieved higher F-measure than those without NEs. All models using NEs improved Recall.

[11]We used an NVIDIA Tesla P100 GPU.

[12]We used an NVIDIA Tesla P100 GPU.

	DC		
	P	R	F1
BERT	0.7402	0.7102	0.7147
BERT$_{FNER}^{BNE}$	0.7375	**0.7142**	**0.7149**
BERT$_{FNER}^{ENE}$	0.7263	**0.7120**	0.7080
BERT$_{NNER}^{BNE}$	0.7348	**0.7128**	**0.7159**
BERT$_{NNER}^{ENE}$	0.7355	**0.7177**	**0.7166**

Table 5: Results of the document classification task. The bold fonts indicate better classification than the baseline BERT. P, R, and F1 denote the precision, recall, and F1-score, respectively.

We see from the results that NEs contribute to improve the score of document classification task with Transformer-based models.

Table 6 shows the results of the headline generation task. We also see that NE information contributed to improved accuracy of the headline generation task. Three models trained using NEs achieved higher accuracy than BART without NEs. The BART$_{NNER}^{BNE}$ showed the same accuracy as BART.

	HG (ROUGE F1)		
	R-1	**R-2**	**R-L**
BART	0.299	0.148	0.265
BART$_{FNER}^{BNE}$	**0.301**	**0.150**	**0.266**
BART$_{FNER}^{ENE}$	**0.301**	**0.149**	**0.266**
BART$_{NNER}^{BNE}$	0.297	$\underline{0.148}$	0.263
BART$_{NNER}^{ENE}$	**0.305**	**0.152**	**0.270**

Table 6: Evaluation results on headline generation. The bold fonts indicate better accuracy than the baseline BERT. The underlined one is the same ROUGE score. R-1, R-2 and R-L indicate ROUGE-1, ROUGE-2 and ROUGE-L, respectively.

BERT$_{NNER}^{ENE}$ showed the best accuracy on document classification. BART$_{NNER}^{ENE}$ also showed the best accuracy on headline generation as in document classification. Our preliminary evaluation shows that NNER exhibited better accuracy than FNER. These results imply that accurate ENE information improved accuracy.

9 Related Work

There are several variants of BERT (Lan et al., 2020; Liu et al., 2019; He et al., 2020). Additionally, recent studies for neural document summarization focus more on pretraining methods (Lewis et al., 2020; Qi et al., 2020; Zhang et al., 2020). Although we used BERT and BART, other methods can combine with the proposed method because the proposed method uses embeddings of NEs from finetuning, which is left as future work.

Marek et al. (2021) proposed an extractive summarization method that uses density of named entities to calculate the importance of a sentence. Furthermore, they proposed an abstractive summarization method that concatenates one-hot representation of named entity categories with token embeddings. Different from Marek et al. (2021), we investigate the effectiveness of the use of NEs in subword-based neural network models and different types of NEs, i.e., BNE and ENE. Additionally, we used embeddings to represent NEs instead of one-hot vectors to obtain further representation.

In addition to document summarization, some works use NEs for improving neural machine translation, which uses similar architectures as neural document summarization. Ugawa et al. (2018) proposed incorporating an additional LSTM layer to encode the sequence of NEs. Li et al. (2018) proposed inserting NE tags into the sequence of words in the source language. Different from Ugawa et al. (2018), we simply used embeddings for encoding NEs because Transformer-based models have self-attention mechanism that uses contextual information to obtain encoded results. Different from Li et al. (2018), our proposed method does not insert NE tags into the sequence of words because the time and memory complexities of Transformer-based models quadratically increase depending on the length of the sequence, which can be a significant problem in headline generation where a source document is long. Additionally, because subword tokenization is not used, their motivation differs from ours that investigates the effectiveness of NEs on subword-based neural networks.

Du et al. (2015) investigated the use of NEs in non-neural network models on document classification. While they reported use of NEs improve the accuracy of document classification, the contribution to subword-based neural network models was not investigated. Pivovarova and Yangarber (2018) compared the representation of NEs for neural network-based models in document classification task. They reported that replacing tokens of named entities with special tokens representing NE categories does not improve the accuracy of document classification. Our experiments showed that combining NE and token embeddings improved the accuracy of document classification.

10 Conclusion

This paper explored the effectiveness of NE information in large-scale pretraining models. We evaluated the effectiveness of NEs in document classification and headline generation tasks. The experimental results showed that NE information improved the accuracy of large-scale SOTA pretraining-based models. We incorporated NE information to pretrained models trained from subword sequences. For future work, we look to explore pretraining methods from subword sequences annotated with NE information from scratch.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual String Embeddings for Sequence Labeling. In Proceedings of COLING 2018, pages 1638–1649.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of NAACL-HLT 2019, pages 4171–4186.

Mian Du, Matthew Pierce, Lidia Pivovarova, and Roman Yangarber. 2015. Improving Supervised Classification Using Information Extraction. In Proceedings of Natural Language Processing and Information Systems, pages 3–18. Springer International Publishing.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. CoRR, abs/2006.03654.

Ryuichiro Higashinaka, Kugatsu Sadamitsu, Kuniko Saito, Toshiro Makino, and Yoshihiro Matsuo. 2012. Creating an Extended Named Entity Dictionary from Wikipedia. In Proceedings of COLING 2012, pages 1163–1178.

Tomoya Iwakura. 2011. A Named Entity Recognition Method using Rules Acquired from Unlabeled Data. In Proceedings of RANLP 2011, pages 170–177.

Taku Kudo and John Richardson. 2018. SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing. In Proceedings of EMNLP 2018, pages 66–71.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In Proceedings of ICLR 2020.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In Proceedings of ACL 2020, pages 7871–7880.

Zhongwei Li, Xuancong Wang, Ai Ti Aw, Eng Siong Chng, and Haizhou Li. 2018. Named-Entity Tagging and Domain Adaptation for Better Customized Translation. In Proceedings of NEWS 2018, pages 41–46.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In Proceedings of Text Summarization Branches Out, pages 74–81.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In Proceedings of ICLR 2019.

Petr Marek, Stepán Müller, Jakub Konrád, Petr Lorenc, Jan Pichl, and Jan Sedivý. 2021. Text Summarization of Czech News Articles Using Named Entities. CoRR, abs/2104.10454.

Lidia Pivovarova and Roman Yangarber. 2018. Comparison of Representations of Named Entities for Document Classification. In Proceedings of RepL4NLP 2018, pages 64–68.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting Future N-gram for Sequence-to-Sequence Pre-training. In Proceedings of EMNLP 2020, pages 2401–2410.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and Korean Voice Search. In Proceedings of ICASSP 2012, pages 5149–5152.

Satoshi Sekine and Hitoshi Isahara. 2000. IREX: IR & IE Evaluation Project in Japanese. In Proceedings of LREC 2000.

Satoshi Sekine and Chikashi Nobata. 2004. Definition, Dictionaries and Tagger for Extended Named Entity Hierarchy. In Proceedings of LREC 2004.

Arata Ugawa, Akihiro Tamura, Takashi Ninomiya, Hiroya Takamura, and Manabu Okumura. 2018. Neural Machine Translation Incorporating Named Entity. In Proceedings of COLING 2018, pages 3240–3250.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. In Proceedings of ICML 2020, volume 119 of Proceedings of Machine Learning Research, pages 11328–11339.

Appendix

A Preprocessing for Data Sets

A.1 Mainichi News Articles

We pre-processed the main texts of Mainichi Shimbun in five steps.

1) We remove articles with no main contents due to copyright.

2) We normalized the texts based on normalization form compatibility composition (NFKC) with `unicodedata.normalize` function of Python 3.6.9.

3) "<" and ">" in normalized texts were replaced by "⟨" and "⟩", respectively. This character-replacing process was needed to distinguish the original and attached characters by attaching the NEC label to text.

4) The texts were attached NEs in XML format with two NE recognizers.

5) We extracted NEs from annotated texts in BRAT[13] format, then we re-attached NEs for the texts because the NE recognizer rarely changed the text.

B Excepted Publication Side Classes

Four classes of `Mai-news` were excluded (i.e., the four classes are front page (01), second page (02), third page (03), and the unknown genre that is not explained (27)) from the 18 classes on only document classification task because the classes were not related to the content of the article.

C Hyper Parameters

Table 7 shows hyperparameters of our experiments on document classification and headline generation.

[13]https://brat.nlplab.org/

Hyper Parameter	**Document Classification**	**Headline Generation**
Optimizer	AdamW (Loshchilov and Hutter, 2019)	
Model Size	BASE	LARGE
Learning Rate	2e-5	3e-5
Batch Size	16	1
Update Freq	16	32
Max Input Token Length	512	1,024
Max Epoch	3	3
Number of Target Classes	14	-
Warm Up Step	0	500
Loss Function	cross entropy	cross entropy
Learning Rate	5e-05	3e-05
Learning Rate Scheduler	linear decay	polynomial decay
Validation Freq.	1,000	20,000
#Max Train Data	-	50,000
#Max Test Data	-	10,000
#Max Dev Data	5,000	3,000
Beam Search Size	-	4
Upper Limit of Length of Token Generation	-	-

Table 7: The hyperparameters in our experiments "#Max Train/Test/Dev Data" show the maximum numbers of data that were used in each experiment ("-" means no upper limit). "Validation Freq." show how many time to process batch size data per one validation in the training phase. The model size of BASE and LARGE indicate that we used BERT BASE and BART LARGE, respectively.