Improving Natural Language Understanding by Reverse Mapping Bytepair Encoding

Chaodong Tong[1,2] **Huailiang Peng**[1,2](✉) **Qiong Dai**[1,2] **Lei Jiang**[1,2] **Jianghua Huang**[3]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]Meituan-Dianping Group, Beijing, China
{tongchaodong, penghuailiang, daiqiong, jianglei}@iie.ac.cn
{huangjianghua}@meituan.com

Abstract

Recently, language models (LMs) or language representation models are widely used in natural language understanding (NLU) tasks. However, these LMs are usually trained on large unlabeled text corpora, while the fine-tuning process simply takes words or word-pieces as model input. Because of the differences between language model and NLU task objectives, the problem of lack of concern on some key words exists. Thus in this paper, we propose a method called reverse mapping bytepair encoding, which maps named-entity information and other word-level linguistic features back to subwords during the encoding procedure of bytepair encoding (BPE). We employ this method to the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018) by adding a weighted linear layer after the embedding layer. We also propose a new model architecture named as the multi-channel separate transformer to evaluate the effectiveness of the newly introduced information by employing a training process without parameter-sharing. Experiments on Story Cloze, RTE, SciTail and SST-2 datasets demonstrate the effectiveness of our approach. Compared with the original results in GPT, our approach gains 1.58% absolute increase on Stories Cloze, 6.4% on RTE, 0.69% on SciTail and 0.8% on SST-2.

1 Introduction

Recently, language models are widely used as the feature extractor for many NLU tasks. Statistical language models learn the joint probability function of sequences of words in a language (Bengio et al., 2003). Trained on large corpora and different domains give LMs generalization abilities, and enable them to capture latent or deep senmantics. Normally, statistical language models take the following objective to maximize:

$$L_1(u) = \sum_i \log P(u_i | u_{i-k}, ..., u_{i-1}; \Theta) \quad (1)$$

where k is the size of the context window, and the conditional probability P is modeled using a neural network with parameters Θ.

However, this formal simplicity determines that they can not deal well with the ambiguity of words nor low-frequency words. In fact, low-frequency words may appear once or little times in the whole corpus, thus the embeddings of them may overfit or underfit the corpus. In practice, we usually truncate them from the vocabulary and replace them with an "UNK" label. In this situation, we partially lose their meanings. For example, in "15 million tonnes of rubbish are produced daily in Cairo.", the number "15" can hardly be trained to gain its proper representation even if it is replaced by a specific label representing numbers. Besides, out of vocabulary (OOV) is a big problem while predicting. Some kinds of word segmentation technologies have been proposed (Joulin et al., 2017; ray Su and yi Lee, 2017) to solve these problems.

BPE (Sennrich et al., 2016) has been proposed to handle these problems as well. It shows its powerful effectiveness in many works (Sennrich and Haddow, 2016; Radford et al., 2018; Devlin et al., 2018). However, it is originally designed to handle open-vocabulary problem in machine translation, basing on the intuition that various word classes are translatable via smaller units than words, for instance names (via character copying or transliteration), compounds (via compositional translation), and cognates and loanwords (via phonological and morphological transformations). Thus compared with semantic characteristics, morphological and compounding characteristics are more considered. The unique meanings of some proper

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 163–173
Hong Kong, China, November 3-4, 2019.

nouns are missing while simply applying it to some NLU tasks. Consider the following textual entailment example:

Example 1. t. *Traditionally, the* ***Brahui*** *of the* ***Raisani*** *tribe are in charge of the law and order situation through the Pass area. This tribe is still living in present day* ***Balochistan*** *in Pakistan.*

h. *The* ***Raisani*** *tribe resides in Pakistan.*

$t \rightarrow h$: *entailment*

BPE:

Brahui: bra hu i

Raisani: rai san i

Balochistan: bal o chi stan

In this example, proper nouns play an important role, but they are divided into wordpieces shared with other words. Especially they have common pieces such as "i" and "o". Therefore, inferring from the encoded sequence can be quite difficult.

Apart from this, let us motivate the lack of concern on some key words or phrases of such method. Here is another example:

Example 2. t. *Cairo is now home to* ***some 15 million*** *people - a burgeoning population that produces* ***approximately 10,000 tonnes*** *of rubbish per day, putting an enormous strain on public services...*

h. ***15 million tonnes*** *of rubbish are produced* ***daily*** *in Cairo.*

$t \rightarrow h$: *not_entailment*

In this case, "15 million tonnes" is the key phrase while "15", "million" and "tonnes" also appear in the context. Word-level or subword-level information is obviously not enough.

To alleviate these issues, we propose a reverse mapping bytepair encoding method to integrate prior knowledge into subwords. The prior knowledge mentioned here includes named-entity information, part-of-speech (POS) tags and dependency parsing labels. Our method has two forms and both of them modify the encoding procedure of BPE. In the conventional form, firstly we tag and parse the target sentence which needs to be tagged with NER, POS taggers as well as a dependency parser. After that, we handle the target sentence with the original BPE algorithm. Note that the taggers and the parser work on the word-level while BPE works on the wordpiece-level. Finally, we encode every wordpiece as a combination of linguistic features of its parent word and itself. To avoid over-reliance on the performance of external tools and error propagation, we modify the former as named-entity phrase based reverse mapping, which adds named-entity phrases to the vocabulary during the scanning process of the whole corpus and encodes a wordpiece as the combination of the named-entity phrase where the wordpiece comes from and itself. We evaluate our method on a set of NLU tasks by applying it to GPT. More specifically, we add a weighted layer after the embedding layer to get different weighted combinations of the inputs. We also propose a new model architecture named as the multi-channel separate transformer to employ a training process without parameter-sharing for wordpieces and additional features. We evaluate our approach on two natural language inference tasks (RTE, SciTail), a question answering task (Story Cloze Test) and a classification task (SST-2), showing the benifits of our approach.

2 Related Work

Improving natural language understanding requires better techniques for modeling natural language. There have been many researchers working on better capturing semantic and morphological information of word vectors (Mikolov et al., 2013a,b; Huang et al., 2012; Levy and Goldberg, 2014b; Pennington et al., 2014). Utilizing internal information has been widely studied, and most of these works employed structural information between words and smaller units (Chen et al., 2015; Iacobacci et al., 2015; Bojanowski et al., 2017; Yu et al., 2017; Xu et al., 2018). Another relative research direction is to use external knowledge. The researchers in Microsoft (Song et al., 2011) employed a big and rich probabilistic knowledge-base to machine learning algorithms, and got significant improvement in terms of tweets clustering accuracy. However, such method needs huge human and material resources to build up a high-quality and extremely wide-coverage knowledge base. Recently, a novel language representation model called ERNIE (Zhang et al., 2019) has been proposed. ERNIE introduces knowledge-related tasks in the pre-training process. Besides, it utilizes graph embedding methods to get the embedded representation of entities. Compared with it, our method does not rely upon external knowledge bases and it can be a double-edged sword. In addition, our method provides syntactic information integration and allows principled integration of named-entity information in an easier way.

There is also a class of method, instead of relying a lot on external knowledge, it takes advantage of the linguistic features that exist in natural language. Levy (Levy and Goldberg, 2014a) generalized the skip-gram model to include arbitrary contexts by dependency parsing. The dependency-based embeddings are less topical and exhibit more functional similarity than the original skip-gram embeddings. Multimodal representations of chinese characters (ray Su and yi Lee, 2017) have also been studied, and the research showed the effectiveness of glyph features in some cases. Sennrich (Sennrich and Haddow, 2016) proposed an approach to employ linguistic features for neural machine translation (NMT). These features include lemmas, subword tags, morphological features, POS tags and dependency labels. Different from us, they paid more attention on how to improve NMT from the morphological level. Besides, they did not consider named entities. Another work relevant to us is (Nallapati et al., 2016). They proposed a feature-rich encoder to capture keywords in summarization. Their model employs a word-level vocabulary, and the words are embedded by concatenating all kinds of features including POS, NER tags and discretized TF and IDF values. Different from them, our approach works on the subword level and leverages linguistic features at the semantic level.

There are many kinds of neural networks that can deal with NLU tasks. Recently, pre-trained language models or language representation models have been widely used and these works got significant improvements (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2018). Trained on large corpora will inevitably face the big-vocabulary problem and the OOV problem. Thus researchers have proposed several methods to handle them. FastText (Joulin et al., 2017) employed n-grams thus it could predict the zero-shot word embeddings. However, n-gram is an arbitrary method of word segmentation. Sennrich (Sennrich et al., 2016) adapted the original byte pair encoding (BPE) compression algorithm to NMT and got significant success. This method iteratively merges most frequent adjacent characters or character sequences in a word until it can not be done, based on the well learned token rank. Therefore it can encode any word with a pre-learned token vocalulary. We give an illustration as Figure 1, showing the ranking process of BPE. GPT (Radford et al., 2018) applied BPE to its training process and got impressive achievements in a series of NLU tasks. Due to the greedy nature of BPE algorithm, the results produced by BPE are deterministic, resulting in insufficient robustness in machine translation. Kudo (Kudo, 2018) located this problem and proposed a subword regularization method to handle it. He trained the NMT model with multiple subword segmentations generated in a probability manner and got improvements especially on low resource and out-of-domain settings. This study provides an interesting insight, but we argue that it may not bring significant improvement in NLU tasks. In fact, BPE is proposed to model open-vocabulary translation in NMT, which is inconsistent with the goal of some natural language understanding tasks. That motivates us to start our work.

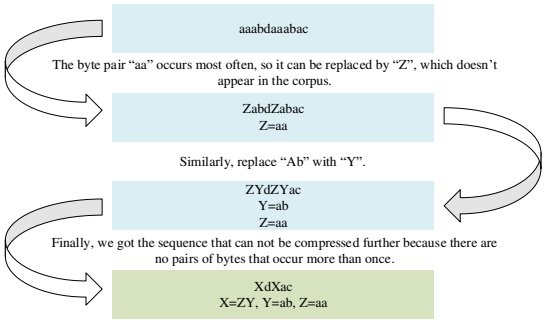

Figure 1: Ranking process of bytepair encoding.

3 Methods

In this section, we introduce our reverse mapping bytepair encoding method and the procedure of applying it to GPT. The reverse mapping bytepair encoding has two forms: label based (LB-BPE) and named-entity phrase based (NPB-BPE). Figure 2 provides a visual illustration. We employ them to the fine-tuning process of GPT by adding a weighted layer after the embedding layer. Besides, we propose a multi-channel separate transformer (MCST) to evaluate the utility of introduced features and give some insights into the semantic capture capabilities of the transformer. The model architecture is shown in Figure 3.

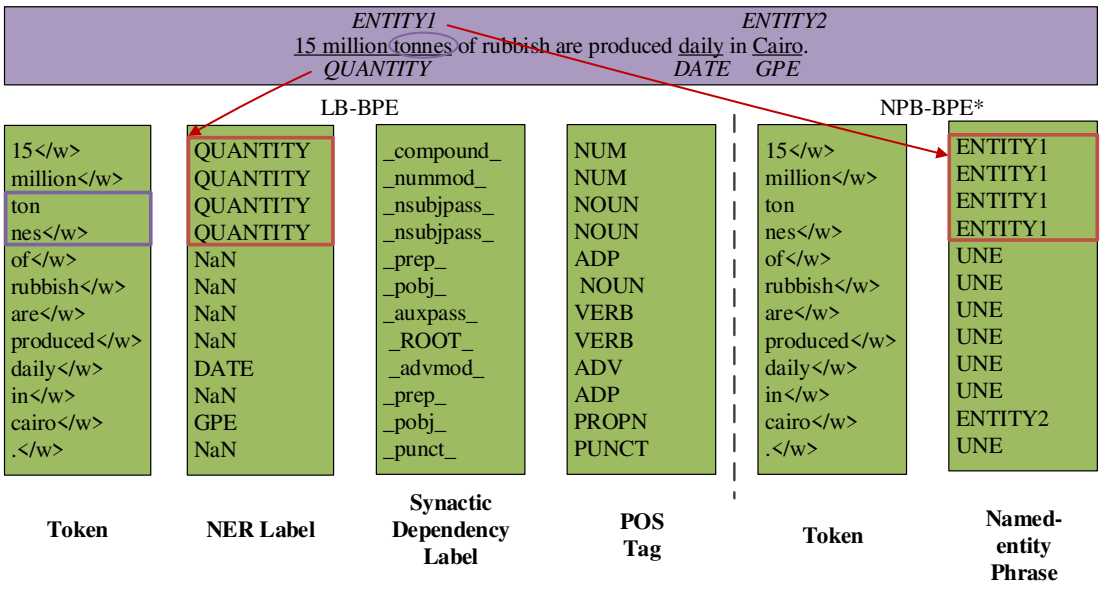

Figure 2: "tonnes" is encoded to be two tokens: "ton", "nes⟨/w⟩". **(Left)** LB-BPE result. Each token consists of four parts: token, NER label, synactic dependency label and POS tag. Words in the original text pass their linguistic features to the tokens generated from them. **(Right)** NPB-BPE* result. Each token consists of two parts: token and corresponding entity.

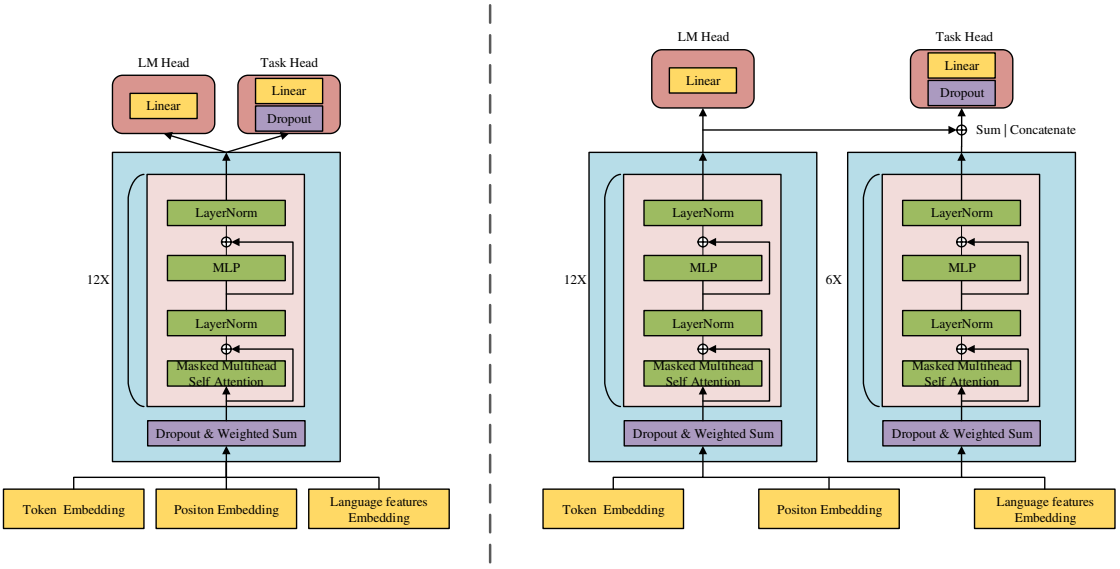

Figure 3: **(Left)** Parameter-sharing training procedure. **(Right)** Multi-channel separate transformer. Language model loss is only generated by the left transformer. The input of the task specific head is the sum or concatenation of both transformers outputs.

3.1 Label Based Reverse Mapping Bytepair Encoding

We extend a token t generated by BPE to four parts $\{t, t_{pos}, t_{ner}, t_{dep}\}$, where t is the same as the original encoded token in BPE, t_{pos} is the POS tag of the word which generates this token, similar with t_{ner} and t_{dep}. Not all words are tagged with named-entity labels, therefore we tag these tokens without NER labels as "NaN". We regard t_{pos}, t_{ner}, t_{dep} as additional features that could not be captured fully by language models. Table 1 shows the amount of features of various types.

Feature type	Amount
NER	20
POS	19
DEP	51

Table 1: Amount of different types of features.

Schemes come from *Spacy Annotation Specifications*[1]. We use *spaCy*[2] to tokenize, tag and parse the datasets during our experiments. We simply use the dependency labels instead of the whole dependency parsing tree because of two reasons: one is it brings complex changes to the input architecture, the other is that we assume the transformer architecture has the ability to establish some kinds of patterns to capure the dependencies between tokens generated by BPE.

3.2 Named-entity Phrase Based Bytepair Encoding

Algorithm 1: Encoding process for NPB-BPE

Input :
Given sentence S;
Pretrained lookup table T for tokens;
Lookup table T_{ne} for named-entity phrases;
NER Scheme Θ := {B,I,O};
Output:
Encoded sequence S';

Current named-entity token array e;
for *each word* $w \in S$ **do**
  Step1. **if** *NER*(w_i) $\in$ {*B,I*} **then**
    Append w_i to e;
    i++;
    Back to Step1;
  end
  else
    e_str := ARR2STR(e)
    if $e\_str \notin T_{ne}$ **then**
      Append e_str to T_{ne};
    end
    for $t' \in$ *BPE*(e_str) **do**
      Append (t', e_str) to S';
    end
    Empty e;
    for $t \in$ *BPE*(w) **do**
      Append (t, "_UNE_") to S';
    end
  end
end

As we mentioned in Introduction 1, some key words are low-frequency. Such as a person name "Kevin Federline", it can be encoded as "kevin⟨/w⟩", "feder" and "line⟨/w⟩", and the meaning will be changed. Passing the keyword information to tokens is a simple and feasible approach to solve this problem. There are many ways of defining which words are key words and thus the problem can be defined as following:

Definition 1. Find an algorithm $A, \forall k \in S$, $A(k) = 1$ if k is a key word, else $A(k) = 0$.

where S represents a sentence.

This is a two-category classification problem. In fact, these key words usually perform as people's names, organizations or other types of named entities. Therefore, we choose a NER algorithm[3] as algorithm A and we modify the encoding process of BPE algorithm as algorithm 1 to pass the entity attribute to tokens. By reverse mapping named-entity words or phrases appeared in the corpus back to BPE tokens, key information has a way to be preserved. Some kinds of named entities (ep. DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL, etc) contain numbers and they usually express general attributes, such as "22-year-old". We add a switch in practice to control whether processing these words with NPB-BPE or not.

The main difference between LB-BPE and NPB-BPE is that LB-BPE makes ordinary use of additional semantic information that can not be easily captured by statistical language models in a specific language, while NPB-BPE provides a way to share important concepts within a corpus.

3.3 Training Process

GPT is followed by many studies for its excellent generalization performance and we will give a brief introduction to it in the first subsubsection. Due to the expensive cost of pre-training, we reuse the OpenAI pre-trained language model[4] parameters, and we train new embeddings and fine-tune all parameters during the fine-tuning process. Figure 4 shows the modified preprocess procedure. Considering the input is enhanced by different kinds of new features, we propose two different processes for training.

Generative Pre-trained Transformer

The Generative Pre-trained Transformer (Radford et al., 2018), also known as OpenAI GPT or called as GPT, is a language representation model which

[1] https://spacy.io/api/annotation#_title
[2] https://spacy.io/
[3] https://spacy.io/api/entityrecognizer
[4] https://github.com/openai/finetune-transformer-lm

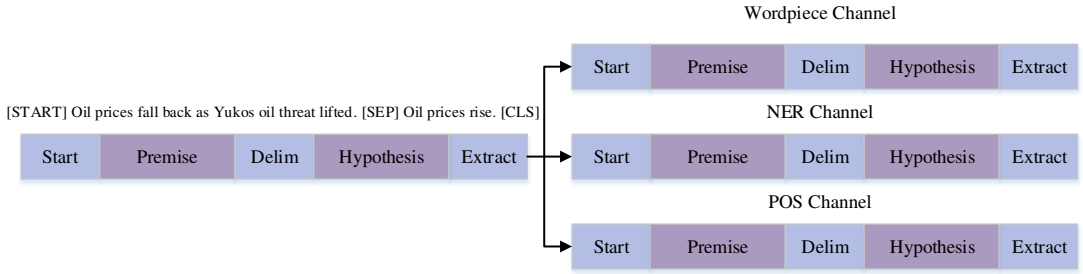

Figure 4: The preprocess procedure of RTE dataset by LB-BPE (NER+POS). Others are similar with this example.

is pre-trained on large corpora and fine-tunes all pre-trained parameters on the downsteam tasks. Its main architechture was originally described in (Vaswani et al., 2017). Unless otherwise stated, the GPT mentioned in this paper refers to a 12-layer decoder-only transformer with masked self-attention heads (768 dimensional states and 12 attention heads). For the position-wise feed-forward networks, we use 3072 dimensional inner states. Other hyperparameters are set as well as the GPT paper (Radford et al., 2018) for comparison.

Sharing Parameters

With this method, we treat additional features like positional encoding. We add these tags to the vocabulary and random initialize their embeddings in the embedding layer, thus the input can be described as following:

$$h_0 = w_{wt}E_t + w_{wp}E_p + \sum_{i \in \mathcal{C}_{lf}} w_{wli}E_{li} \quad (2)$$

where E_t, E_p, E_{li} represent the token embedding matrix, position embedding matrix and linguistic feature embedding matrix, $\mathcal{C}_l$ is the collection of all types of linguistic features and w_{wt}, w_{wp}, w_{wli} are the corresponding weight scalars. The following data flow is the same as GPT (Liu et al., 2018). We use the following objective to minimize:

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda L_1(\mathcal{C}) \quad (3)$$

where $\mathcal{C}$ represents the labeled dataset, $L_2(\mathcal{C})$ is the cross entropy loss of classification got on $\mathcal{C}$, $L_1(\mathcal{C})$ is the cross entropy loss of the language model got on $\mathcal{C}$ and λ is a weight parameter between 0 and 1.

Multi-channel Separate Transformer

GPT uses a multi-layer transformer decoder as the language model, and it learns patterns in the language by simply observing the token-level sequences. In this part, we use LB-BPE to encode sequences and with this method we have multiple input channels. We employ two stand-alone multi-layer transformer decoder to separate the parameters learned on different perspectives. One is the 12-layer pre-trained transformer released by OpenAI GPT, and the other is an entirely new one which has different amount of layers with the former and takes responsibility for linguistic features.

4 Experiments and Analysis

4.1 Setup

We follow the model hyperparameters mentioned in the GPT (Liu et al., 2018) paper. We use learned position embeddings with variable length depending on downstream tasks. We perform a pilot experiment and the result shows that it is almost the best to set the weighted layer as 1.0 for each channel. We use a 16G *ASPEED Graphics Family (rev 30)* card for training.

4.2 Supervised Fine-tuning

Datasets

Story Cloze Test Story Cloze Test (Mostafazadeh et al., 2017) is a new commonsense reasoning framework for evaluating story understanding, story generation, and script learning. This test requires a system to choose the correct ending to a four-sentence story. We choose it as a part of our evaluation corpora because it requires the model to capture rich linguistic phenomena.

RTE and SciTail RTE (Bentivogli et al., 2009) and SciTail (Khot et al., 2018) are language in-

Model	Story Cloze(Acc%)	RTE(Acc%)	SciTail(Acc%)	SST-2(Acc%)
jose.fonollosa's model	87.60	-	-	-
BiLSTM+ELMo+Attn	-	58.9	-	90.4
BigBird	-	-	**93.84**	-
GPT	86.5	56.0	88.3	91.3
NER	**87.39**	**61.6**	**88.90**	91.2
NER+POS	**88.08**	**62.1**	**88.33**	91.1
NER+DEP	**87.55**	**62.0**	**88.99**	90.2
NER+POS+DEP	**87.07**	**60.4**	87.54	**92.1**
POS	84.07	**62.4**	88.15	**92.1**
POS+DEP	86.05	**61.5**	87.63	**91.7**
DEP	86.48	**62.0**	87.82	91.3
NPB-BPE	**87.76**	**58.8**	**88.43**	91.1
NPB-BPE*	**87.39**	**62.3**	87.16	**91.9**

The details of the models for comparison in this table can be found in the leaderboards of the corresponding dataset official websites. Each row in the second block represents a LB-BPE result with different combination of features. NER represents using named-entity labels, POS represents part of speech tags and DEP represents synactic dependency parsing tags. NPB-BPE* represents filtering out named entities within these types (DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL).

Table 2: Reverse mapping bytepair encoding results on Story Cloze Test, RTE, SciTail and SST-2.

ference also called textual entailment tasks which given a text t and a hypothesis h, t entails h, if, typically, a human reading t would infer that h is most likely true. The relation is directional. These two datasets differ from each other in size and domain.

SST-2 SST-2 (Socher et al., 2013) is related to sentiment analysis, containing 56.4k movie reviews and each of them has a binary label.

Evaluation of Label Based Reverse Mapping Bytepair Encoding

We evaluate LB-BPE on these datatsets and conduct multiple controlled trials based on different combinations of linguistic features by employing the parameter-sharing training procedure. Details is described in Table 2. The result shows that incorporating named-entity features usually works well, while utilizing POS tags sometimes can bring significant performance improvement. Besides, it is not a good way to use external features as many as possible, most likely because of the error propagation. However, combinations of named-entity information and others may bring a small boost. Compared with GPT, we achieved an absolute increase of 1.58% on Story Cloze Test, 6.4% on RTE, 0.69% on SciTail and 0.8% on SST-2 at our best.

Evaluation of Named-entity Phrase Based Bytepair Encoding

We evaluate NPB-BPE with the parameter-sharing training procedure. Details are shown in Table 2. Overall, NPB-BPE improves the performances for all datasets while not overly relying on external features. On Story Cloze Test, both kinds of NPB-BPE are not worse than LB-BPE (NER). On RTE, NPB-BPE* nearly achieves the best performance of LB-BPE. Little performance boost is shown on SciTail, probably because sentences in SciTail are more focused on the expressions of concepts and knowledge, and thus key words are more about verbs and nouns rather than named entities. However, as we unexpected, applying NPB-BPE* to SST-2 gains a 0.6% absolute increase. We also collect the statistics of named entities on these datasets. Details is shown in Table 3. We observe that there is a positive correlation between named entities percentage and performance improvement.

Evaluation of Different Training Processes

The parameters in GPT are all the same for all wordpiece embeddings in the vocabulary. However, linguistic features contain different levels of information compared with words and wordpieces. Thus we evaluate two different training processes mentioned in Section 3.3. As for MCST,

Dataset	Amount	Percentage(%)	Top10(non-numeric)
Story Cloze	2354	7.3	(“One day”, 241), (“John”, 196), (“Amy”, 195), (“Bob”, 189), (“Joe”, 156), (“Tom”, 152), (“Tim”, 148), (“Sam”, 114), (“Gina”, 101), (“Jim”, 95)
RTE	15548	21.9	(“U.S.”, 294), (“Iraq”, 262), (“US”, 243), (“the United States”, 180), (“China”, 166), (“today”, 150), (“France”, 148), (“Monday”, 144), (“American”, 144), (“Bush”, 142)
SciTail	7631	4.8	(“Earth”, 670), (“earth”, 632), (“Sun”, 358), (“Mercury”, 262), (“Oxygen”, 139), (“Hydrogen”, 134), (“Venus”, 103), (“Jupiter”, 101), (“Weight”, 101), (“Mars”, 92)
SST-2	728	1.6	(“american”, 141), (“years”, 128), (“summer”, 122), (“the year”, 119), (“year”, 90), (“today”, 87), (“2002”, 60), (“this year”, 54), (“two hours”, 53), (“saturday”, 51)

Table 3: Statistics of named entities on Story Cloze Test, RTE, SciTail, SST-2.

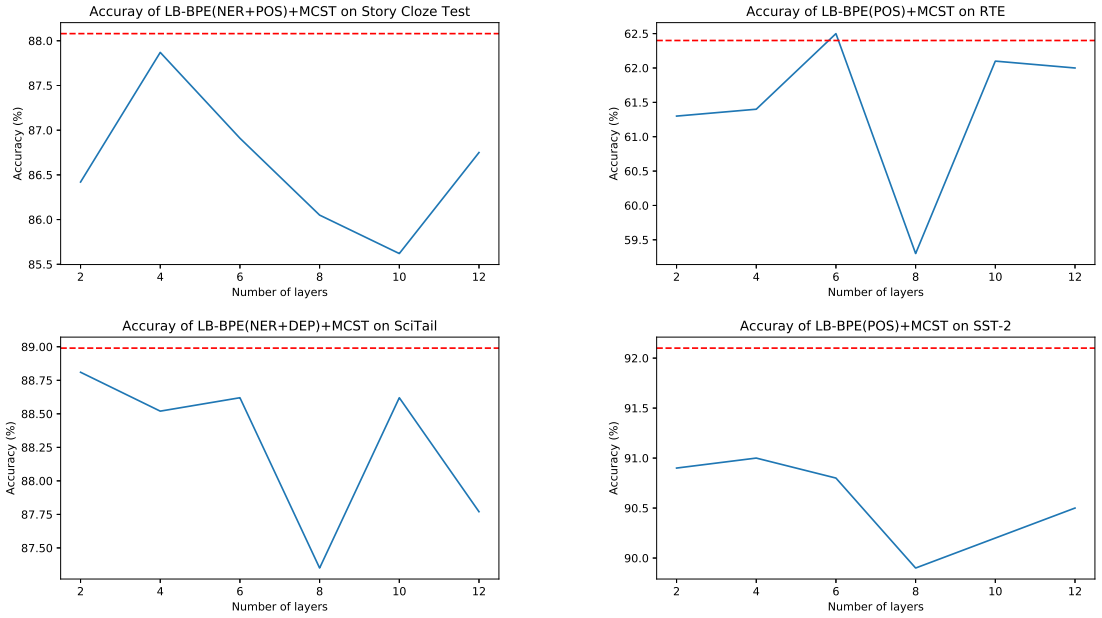

Figure 5: Accuracy of LB-BPE+MCST on all datasets. The red dotted line in each subgraph represents the best result of LB-BPE+GPT got on that dataset.

we perform a series of experiments based on different amount of blocks in the new transformer. As shown in Figure 5, employing non-parameter-sharing training process will generally reduce the performance. We infer that it is probably due to the lack of pre-training procedure for newly introduced features. However, MCST predicts labels based on both transformers while the accuracy rates do not drop too much, which suggests that the newly introduced linguistic features work at least to some extent and the transformer architecture can capture some potential semantic information from them independently. Besides, MCSTs trained with 2, 4, 6 layers in the new transformer perform best in our experiments, and they gain similar or even better results compared with the parameter-sharing training process, which means training newly introduced features doesn't need as many parameters as the original transformer. The main reason is that linguistic features act at a higher level compared with words and wordpieces.

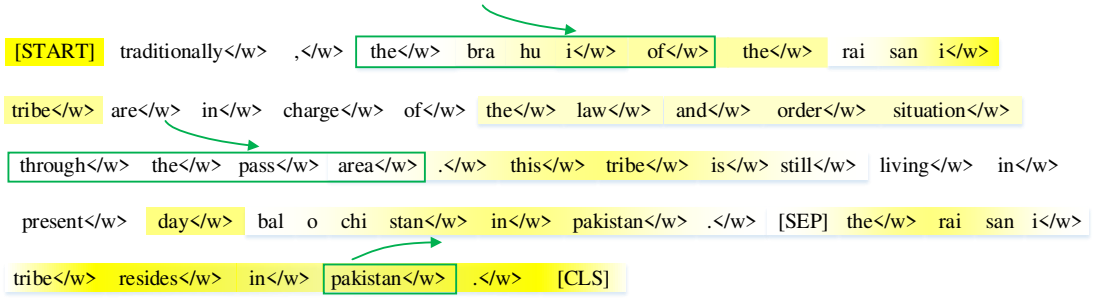

Figure 6: Attention weights of a textual entailment case in RTE. Yellow parts represent the attention weights of our approach. Green arrows and boxes represent the weight changes compared with GPT.

4.3 Case Study

We compare the attention weights of GPT and our LB-BPE (NER+POS) approach in a textual intailment example as shown in Figure 6. GPT labels it as *not_entailment* while our approach labels it as *entailment* which is the right answer. The attention weights come from the same attention head in the last transformer block. Changes in some parts lead to the correct answer.

5 Conclusion and Future Work

In this paper, we introduce a simple approach called reverse mapping bytepair encoding to improve natural language understanding based on pre-trained language models. The reverse mapping bytepair encoding has two forms: One is label based and the other is named-entity phrase based. Both forms introduce extra information to the tokens generated by BPE while the former ordinarily employs linguistic features, and the latter provides a way to share concepts through named-entity phrases. In addition, in the second form, we summarize the problem we are trying to solve into a two-category problem that judges whether a word is a key word. By applying them to the fine-tuning process of GPT, we gain about 1-6% improvement on downsteam NLU tasks. We also propose a new model architecture named as MCST and experiments based on it shows its effectiveness in some cases. Besides, the experimental results show that linguistic features usually perform at a higher level compared with words and wordpieces. It is quite easy to apply our method to the existing language models.

There could be several directions to be explored for future works. Language models have many forms, we only test our approach on GPT, a follow up direction is finding if it is generic enough. In this paper, we don't employ pre-training for linguistic features, it might be better by doing this. There are several researches focusing on incorporating knowledge into systems to improve their performances, thus we are looking forward to finding a smooth way to utilize named entities with prior knowledge or knowledge graphs.

Acknowledgments

This work is supported by National Key Research and Development Program of China under Grant No. 2017YFB0803003 and No. 2016YFB0801302. We also thank the anonymous reviewers and the area chairs for their valuable comments and suggestions that help improve the quality of this manuscript.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo. Magnini. 2009. The fifth PASCAL recognizing textual entailment challenge.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan. 2015. Joint learning of character and word embeddings. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. 2017. LSDSem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Yangqiu Song, Haixun Wang, Zhongyuan Wang, Hongsong Li, and Weizhu Chen. 2011. Short text conceptualization using a probabilistic knowledgebase. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Tzu ray Su and Hung yi Lee. 2017. Learning chinese word representations from glyphs of characters. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Yang Xu, Jiawei Liu, Wei Yang, and Liusheng Huang. 2018. Incorporating latent meanings of morphological compositions to enhance word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Jinxing Yu, Xun Jian, Hao Xin, and Yangqiu Song. 2017. Joint embeddings of chinese words, characters, and fine-grained subcharacter components. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.