electronics

Article

Part-of-Speech Tagging with Rule-Based Data Preprocessing and Transformer

Hongwei Li , Hongyan Mao * and Jingzi Wang

Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China; 51194501008@stu.ecnu.edu.cn (H.L.); jingziwang@163.com (J.W.)
* Correspondence: hymao@sei.ecnu.edu.cn

Abstract: Part-of-Speech (POS) tagging is one of the most important tasks in the field of natural language processing (NLP). POS tagging for a word depends not only on the word itself but also on its position, its surrounding words, and their POS tags. POS tagging can be an upstream task for other NLP tasks, further improving their performance. Therefore, it is important to improve the accuracy of POS tagging. In POS tagging, bidirectional Long Short-Term Memory (Bi-LSTM) is commonly used and achieves good performance. However, Bi-LSTM is not as powerful as Transformer in leveraging contextual information, since Bi-LSTM simply concatenates the contextual information from left-to-right and right-to-left. In this study, we propose a novel approach for POS tagging to improve the accuracy. For each token, all possible POS tags are obtained without considering context, and then rules are applied to prune out these possible POS tags, which we call rule-based data preprocessing. In this way, the number of possible POS tags of most tokens can be reduced to one, and they are considered to be correctly tagged. Finally, POS tags of the remaining tokens are masked, and a model based on Transformer is used to only predict the masked POS tags, which enables it to leverage bidirectional contexts. Our experimental result shows that our approach leads to better performance than other methods using Bi-LSTM.

Keywords: Part-of-Speech tagging; NLP; Transformer; rule-based data preprocessing

check for updates

Citation: Li, H.; Mao, H.; Wang, J. Part-of-Speech Tagging with Rule-Based Data Preprocessing and Transformer. *Electronics* **2022**, *11*, 56. https://doi.org/10.3390/electronics11010056

Academic Editor: Chiman Kwan

Received: 13 October 2021
Accepted: 21 December 2021
Published: 24 December 2021

1. Introduction

Part-of-Speech (POS) tagging is one of the most important tasks in the field of natural language processing (NLP). It assigns a POS tag to each word in a given sentence. For a short and simple sentence "*I like dogs*", a POS tagger can easily identify the word *I* as a pronoun, the word *like* as a verb, and the word *dogs* as a noun. However, some words in complex sentences are difficult to tag correctly by POS taggers. The same word in a different context has different POS tags, which makes POS tagging a challenging task.

POS tagging can be an upstream task for other NLP tasks, such as semantic parsing [1], machine translation [2], and relation extraction [3], to improve their performance. Hence, improving the accuracy of POS tagging becomes an important goal.

For example, the dependency parser in Stanza pipeline [4] takes the result of POS tagging as part of the input because POS tagging is helpful for dependency parsing [5]. Although current POS taggers have achieved 97.3% token accuracy, the sentence accuracy is not as high [6]. This may cause performance loss for the dependency parser because it utilizes POS tags of all tokens to extract a dependency parse tree of a sentence. It is the wrong POS tagging for one word that may result in the extraction of a wrong tree.

In recent years, most POS taggers have used bidirectional Long Short-Term Memory (Bi-LSTM) [7,8] for POS tagging. In addition to word-level embeddings, they append other types of embeddings to improve the accuracy. However, Bi-LSTM is not as powerful as Transformer [9] in leveraging contextual information, since Bi-LSTM simply concatenates contextual information from left-to-right and right-to-left. With the self-attention mechanism, deep learning models based on Transformer may deliver performance gains for POS tagging.

In this paper, we propose a novel approach to improve the accuracy of POS tagging, which includes rule-based data preprocessing and a deep learning model. Figure 1 shows an example of POS tagging with our approach. During the rule-based data preprocessing, for each token, all possible POS tags are obtained without considering the context. Then, rules are applied to prune out these possible POS tags. After pruning, most of the tokens only possess one candidate POS tag, and they are considered to be correctly tagged by the data preprocessing. In the inference phase, POS tags of the remaining tokens are masked, and a deep learning model is responsible for predicting the masked POS tags. The model is based on the Transformer's encoder, and a POS (Part-of-Speech) embedding layer is introduced to accept POS tags tagged by the data preprocessing, which is helpful in predicting the POS tags of the remaining tokens. By combining the rule-based data preprocessing and deep learning, we obtain POS tags of all tokens.

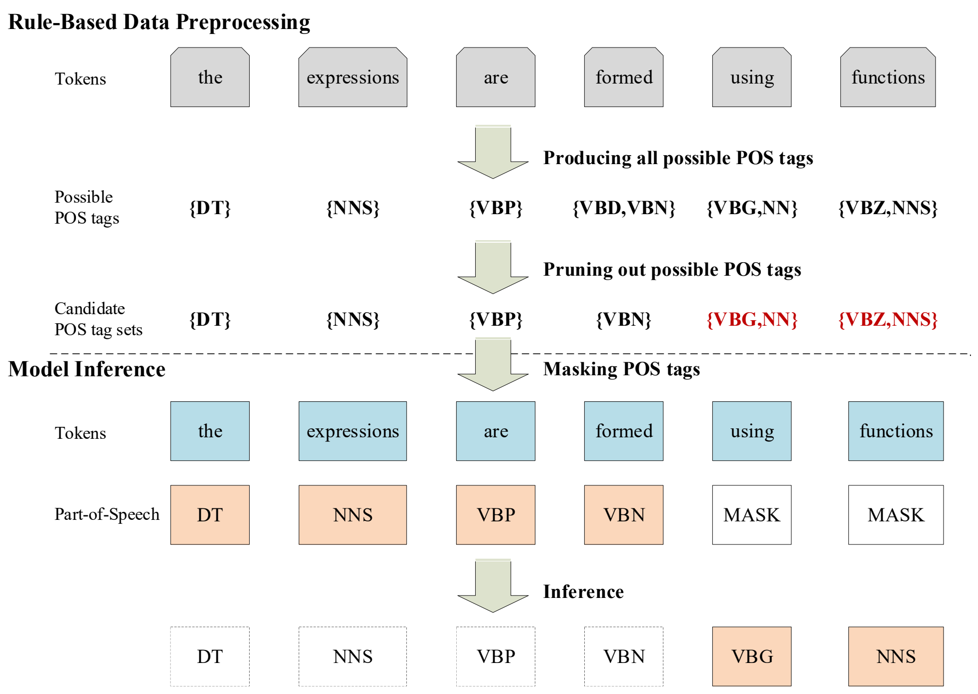

Figure 1. POS tagging for the sentence: *"The expressions are formed using functions"*. In the step of pruning out possible POS tags, the POS tag VBD of the token *formed* is eliminated after one of our rules is used to tag the verb following the token *are* as VBN or VBG. For the token *using* and *functions*, their POS tags cannot be determined and are therefore masked. In the inference phase, the masked POS tags are predicted by a deep learning model.

The contributions can be summarized as follows:

(1) To further improve the accuracy of POS tagging, we propose a novel approach for POS tagging, which combines the rule-based methods and deep learning.
(2) We implement a rule-based method to tag some portion of the words. It can enhance the performance of POS tagging when combined with deep learning.
(3) The proposed method utilizes the self-attention to capture dependencies between words at any distance. Moreover, we mask a certain portion of POS tags and the model only predicts the masked POS tags, which enables the model to better exploit the global contextual information.
(4) We evaluate our method on a public dataset. On the dataset, the method achieves a per-token tag accuracy of 98.6% and a whole-sentence correct rate of 76.04%. Experimental results demonstrate the effectiveness of the method.

This paper is organized as follows. In the next section, we review related work. Section 3 provides details of the rule-based data preprocessing. Section 4 presents the structure of the deep learning model. Section 5 gives the experimental settings and evaluation. Finally, we conclude the paper with a summary in Section 6 and give an outlook on future work in Section 7.

2. Related Work

In this section, we review the related work on the Penn Treebank POS tagset, POS tagging, and Transformer, respectively.

2.1. Penn Treebank Tagset

The Penn Treebank POS tagset [10], which contains 36 POS tags and 12 other tags, is widely used to annotate large corpora of English (See Table 1). To tag words in a given sentence with specific POS tags, the Penn Treebank POS tagset is adopted in this paper.

Table 1. The Penn Treebank POS tagset.

POS Tag	Description	POS Tag	Description
CC	Coordinating conjunction	TO	*to*
CD	Cardinal number	UH	Interjection
DT	Determiner	VB	Verb, base form
EX	Existential *there*	VBD	Verb, past tense
FW	Foreign word	VBG	Verb, gerund/present participle
IN	Preposition	VBN	Verb, past participle
JJ	Adjective	VBP	Verb, non-3rd
JJR	Adjective, comparative	VBZ	Verb, 3rd
JJS	Adjective, superlative	WDT	*wh*-determiner
LS	List item marker	WP	*wh*-pronoun
MD	Modal	WP$	Possessive *wh*-pronoun
NN	Noun, singular or mass	WRB	*wh*-adverb
NNS	Noun, plural	#	Pound sign
NNP	Proper noun, singular	$	Dollar sign
NNPS	Proper noun, plural	.	Sentence-final punctuation
PDT	Predeterminer	,	Comma
POS	Possessive ending	:	Colon, semi-colon
PRP	Personal pronoun	(	Left bracket character
PRP$	Possessive pronoun	)	Right bracket character
RB	Adverb	“	Straight double quote
RBR	Adverb, comparative	‘	Left open single quote
RBS	Adverb, superlative	“	Left open double quote
RP	Particle	’	Right close single quote
SYM	Symbol	”	Right close double quote

2.2. POS Tagging

There exist different methods for POS tagging, such as rule-based methods, methods based on linear statistic models, and deep learning methods based on Bi-LSTM.

Brill [11,12] proposes a trainable rule-based POS tagger, which can automatically construct rules and use the rules to tag all tokens in a given sentence. However, this is difficult to use on real data due to the complexity of natural languages. Some works are based on linear statistic models, such as Conditional Random Fields (CRF) [13] and Hidden Markov [14]. These statistic models perform relatively well on the corpora tagged with a coarse-grained tagset, but they do not perform as well as the Bi-LSTM on the corpora tagged with a fine-grained tagset [15].

In recent years, methods for POS tagging have mainly been based on Bi-LSTM since this is a powerful model that captures time dynamics via recurrence [16]. There are several methods for learning the vector representation of words, such as Word2Vec [17], fastText [18], and Glove [19]. Bi-LSTM takes the vector representations as input and leverages the semantic information from the representations to assign a POS tag to each element.

Wang et al. [20] use Bi-LSTM for POS tagging. In addition to the word embedding layer, a function is introduced to indicate the original case of words. Ling et al. [21] propose a C2W model based on LSTM, which composes representations of characters into representations of words. The experimental result shows that the C2W model achieves better performance than the word lookup tables in POS tagging. Plank et al. [22] also use Bi-LSTM as a base model for POS tagging, where the input includes not only word-level embeddings but also the character-level embeddings. The POS tagger in Stanza pipeline [4] adopts a highway Bi-LSTM [23] with inputs coming from the concatenation of three sources: (1) a pretrained word embedding; (2) a trainable frequent word embedding; (3) a character-level embedding [24], and uses affine classifiers for each type of tag [25]. The above methods [4,20–22] improve the accuracy of POS tagging by enriching the input information. To further improve the accuracy of POS tagging, some works [26–28] combine Bi-LSTM with CRF since CRF can learn sentence-level tag information. In addition to CRF, Bi-LSTM can be integrated with adversarial neural networks to extract better features [29,30], which also can improve the accuracy. POS tagging with Bi-LSTM typically requires a large number of annotated samples. To solve the problem of lacking a large number of training samples, some works [31–33] apply transfer learning to POS tagging.

2.3. Transformer

Vaswani et al. [9] propose a new network architecture different from recurrent neural networks (RNNs) and LSTMs, Transformer, which is solely based on the self-attention mechanisms and eschews recurrence and convolutions.

The self-attention mechanism in Transformer enables it to efficiently capture dependencies between words at any distance. The input to the self-attention function is composed of queries, keys of dimension d_k, and values. The queries, the keys, and the values are mapped into three representations Q, K, and V with three linear layers, then the attention is computed on Q, K, and V.

$$Attention(Q,\ K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

For NLP, Transformer architecture has become the de-facto standard [34] thanks to the self-attention mechanism. In particular, some pretrained language models based on Transformer, such as BERT [35] and its variant RoBERTa [36], have achieved state-of-the-art results on different NLP tasks. They belong to masked language modeling, which is more powerful than standard conditional language models in utilizing both left-to-right and right-to-left contextual information.

Considering the power of the Transformer, we propose to build a model for POS tagging based on Transformer.

3. Rule-Based Data Preprocessing

The rule-based data preprocessing can acquire POS tags of most of the tokens, which consists of three steps. The first step is producing all possible POS tags of each token. This is followed by the pruning, where the context is considered to reduce the number of candidate POS tags of each token. The last step is masking POS tags that are prepared for model training or inference.

3.1. Producing All Possible POS Tags

Without considering the context, all possible POS tags of each token are acquired through the process of lemmatization and transformation. After the two processes, the POS tag in a given context is assured to be in the possible POS tags.

In the process of lemmatization, each token is lemmatized based on simple word deformation rules. Specifically, each token is converted into lemmas by modifying its suffixes. Then, a dictionary containing only lemmas and their basic POS tags (See Table 2) is used to check whether the lemmas are correct.

Table 2. The dictionary.

Lemmas	Basic POS Tags
watch	NN, VB
small	RB, JJ, NN
and	CC
the	DT
his	PRP$
him	PRP
to	TO, IN
at	IN
...	...

For instance, the process of lemmatization for the token *watches* is as follows. Firstly, we check whether the token itself is a lemma. According to the dictionary, the token *watches* cannot be a lemma. Secondly, various possible ways of editing suffixes are tried, such as deleting the suffix *s* to acquire the token *watche*, but the token *watche* does not exist in the dictionary and, thus, cannot be the lemma of the token *watches*. The only way to get its lemma *watch* is to delete the suffix *es*. Finally, the dictionary is queried for basic POS tags. For the lemma *watch*, its basic POS tags are NN and VB.

In the process of transformation, lemmas with different basic POS tags are reverted to the token, which is also based on the deformation rules. Meanwhile, possible POS tags of the token can be acquired.

For the lemma *watch* with the POS tag NN, only by adding the suffix *es* can the lemma *watch* be transformed to the token *watches*, and its POS tag is identified as NNS. For the lemma *watch* with the POS tag VB, we can get the token *watches* with the POS tag VBZ. After the above processes, the possible POS tags for the token *watches* are NNS and VBZ.

However, there are a small number of words whose possible POS tags cannot be acquired in the above way, including words with irregular deformations. As shown in Table 3, their possible POS tags are cached so that they can be obtained directly without the above process.

Table 3. The cached POS tags.

Words	Possible POS Tags
stopped	VBD, VBN
better	JJR, RBR, VB, VBP
cast	VB, VBP, VBD, VBN, NN
children	NNS
...	...

It is possible to obtain wrong lemmas and, thus, obtain impossible POS tags because of the simple method, but this is very rare. Even if impossible POS tags are obtained, it does not matter, since the impossible POS tags can be filtered out in the next step. Furthermore, the deep learning model is used to predict its POS tag if the impossible POS tags cannot be filtered out.

3.2. Pruning out Possible POS Tags

Once all possible POS tags of each token are obtained, rules can be applied to prune out the possible POS tags, from which some POS tags are excluded or selected.

It is almost impossible to rely entirely on rule-based methods to correctly label all tokens. Even if it is possible, an enormous number of rules are required, and they may conflict with each other, which causes the algorithm to be time-consuming. Therefore, a compromise solution is adopted, where rare cases are ignored to reduce the number of rules. Instead of using rule-based methods to tag all tokens, our aim is to tag some portion of the tokens.

The object of POS tagging is mostly declarative sentences. The declarative sentences must meet the principle that there is at least one finite verb except for elliptical sentences. Hence, the POS tags of some verbs can be directly determined, such as the word *am*, the word *does*, the word *have*, modal verbs and their inflections, and we start with these verbs to tag tokens behind them. For instance, in the sentence fragment *"has been redecorated"*, the word *has* is tagged with VBZ, and then the word *been* is tagged with VBN. At last, the word *redecorated* is tagged as VBN because of the word *been*.

The POS tag of a word is constrained by the POS tags of its surrounding words. As long as the POS tag of a word is determined, it can be used for reducing the candidate POS tags of the surrounding words.

In the previous step, there are words that have only one possible POS tag, such as DT, IN, CC, PRP, or PRP$. In most cases, these POS tags do not appear in the candidate POS tags of words with other POS tags. Therefore, the rules mainly focus on other POS tags.

For the word that has multiple candidate POS tags, these tags are eliminated when:

- It follows a preposition or determiner, and these tags are VB, VBP, VBD, VBZ, and MD.
- It follows an adjective, and these tags are RB, RBR, RBS, VB, VBP, VBD, VBZ, and MD.
- It is followed by an adverb, and these tags are JJ, JJR, and JJS.
- It is followed by or follows a verb with the POS tag VB, VBP, VBD, VBZ, or MD, and these tags are VB, VBP, VBD, VBZ, and MD.

There are also several rules for selection from the candidate POS tags as follows:

- If it follows a preposition or determiner, or there are modifiers between the word and the preposition or the determiner, and the word can be used as a noun but cannot be used as a modifier, then the word is tagged NN or NNS.
- If it is followed by a noun and its candidate POS tags contains JJ, JJR, JJS, VBN, or VBG, then these POS tags are selected as new candidate POS tags.
- If it is followed by an adjective and its candidate POS tags contains RB, RBR, RBS, VB, VBP, VBD, VBN, or VBZ, then these POS tags are selected as new candidate POS tags.

For the word whose candidate POS tags only contain VB and VBP, its POS tag is identified as VB when:

- It is the first word in a sentence.
- It follows an adverb that is the first word in a sentence.
- It follows the word *to* or there are adverbs between it and the word *to*.

For the word *to*, it is necessary to make a distinction between the POS tag TO and IN. Three situations are simply divided: (1) If it is followed by a verb with the POS tag VB, its POS tag is identified as TO. (2) If it is followed by a noun, an adjective, or a verb with the POS tag VBG or VBN, its POS tag is identified as IN. (3) If it is followed by an adverb, the POS tag of the word following the adverb needs to be observed. If the word is an adjective, its POS tag is identified as IN. Otherwise, its POS tag is identified as TO.

We have constructed the above rules, which can cover the vast majority of cases. In addition to the construction of the rules, the order in which the rules are applied is critical. For example, in the sentence *"He did make it"*, the fourth rule in the rules for excluding POS tags is not applicable if no other rule is considered. However, if the rule to tag the token *did* as VBD and then tag the token *make* as VB is applied before the fourth rule, there is no problem with the fourth rule. Since the number of the above rules is small, we can manually adjust the order to achieve a best performance.

The rules are iteratively applied to prune out the POS tags until candidate POS tags of each word do not change. Algorithm 1 presents the pseudo-code of the pruning. We abstract the application of each rule as a process called *ApplyRule*. In the process *ApplyRule(rule, words, sets)*, the rule denoted by the variable *rule* is applied to each word in the variable *words* to prune out its candidate POS tags. For each word, its local contextual information, such as its candidate POS tagset, its surrounding words, the POS tags of the surrounding words, and the positions of the surrounding words, are accessed to determine if the conditions of the rule are satisfied. If satisfied, some POS tags are excluded or selected from the candidate POS tagset of the word according to the rule. Otherwise, the rule processes the next word. After the rule is applied to all words, the process *ApplyRule(rule, words, sets)* returns a Boolean value that indicates whether there exists a POS tagset whose content changes relative to the content before the rule is applied. If the return value is true, the variable *flag* will be updated to true and, thus, the variable *changed* will also be updated to true. This causes a new iteration to be performed until the variable *changed* is equal to false.

Algorithm 1: The pruning

```
Input: The rules rules, all the words in a sentence words, and their candidate POS tagsets sets
changed = TRUE
while changed == TRUE
  flag = FALSE
  for i = 1 to rules.length
    rule = rules[i]
    flag = (flag || ApplyRule(rule, words, sets))
  changed = flag
```

In most cases, the number of iterations through all rules (the number of iterations of the while loop in the above pseudo-code) is not more than five. Therefore, the pruning is not time-consuming.

After pruning, the candidate POS tagset of each token is obtained. If the candidate POS tagset of a token contains one POS tag, the token is considered to be correctly tagged. Otherwise, the POS tag of the token is predicted by deep learning models. In most instances, POS tags of most tokens (about 68% on the dataset mentioned in Section 5.1) can be determined. Ideally, POS tags of all tokens in a sentence are tagged correctly, which happens in simple and short sentences.

3.3. Masking POS Tags

There are two most successful pretraining objectives, autoregressive language modeling and autoencoding [37]. BERT is based on denoising autoencoding, which masks 15% of all tokens at random and only predicts the mask tokens [35]. This allows BERT to utilize bidirectional contexts, which is more powerful than the shallow concatenation of a left-to-right and a right-to-left model.

Inspired by the idea of BERT, we mask a certain portion of POS tags and only predict the masked POS tags. Figure 2 illustrates an example. Unlike random masking in BERT, the POS tag of the token is masked if the candidate POS tagset of a token contains more than one POS tag.

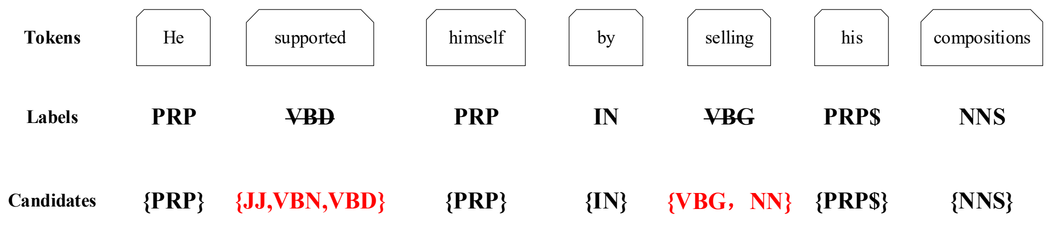

Figure 2. Masking partial POS tags. The POS tags of the token *supported* and the token *selling* are masked.

In this manner, our model is able to exploit POS tags of its surrounding tokens to predict the POS tag of the token if the POS tag of a token is masked. Moreover, the model focuses on learning how to label tokens whose POS tags are difficult to obtain through the data preprocessing.

4. Tagging with Transformer

Given a sentence $w_1, w_2, \ldots, w_N$ with POS tags $y_1, y_2, \ldots, y_N$ and $m_1,\ m_2, \ldots\ , m_N$. Our deep learning model aims to predict the POS tag probability distribution. Here, $m_i = 1$ indicates y_i is masked and $m_i = 0$ indicates y_i is not masked.

4.1. Model

Our model uses the Transformer's encoder as a base model. With the masking, the encoder is able to make better use of bidirectional contexts than the Bi-LSTM that simply concatenates contextual information from left-to-right and right-to-left.

The input layer is shown in Figure 3, which is composed of the word embedding layer, the POS embedding layer, and the position embedding layer. The word embedding is a vectorized representation of words. Similarly, the POS embedding represents specific POS tags. As shown in Figure 4, for tokens whose POS tags are masked, their POS embeddings are replaced with $\mathrm{E}_{\mathrm{[MASK]}}$. In addition to word embeddings and POS embeddings, position embeddings are required to indicate the position, because Transformer eliminates recurrence.

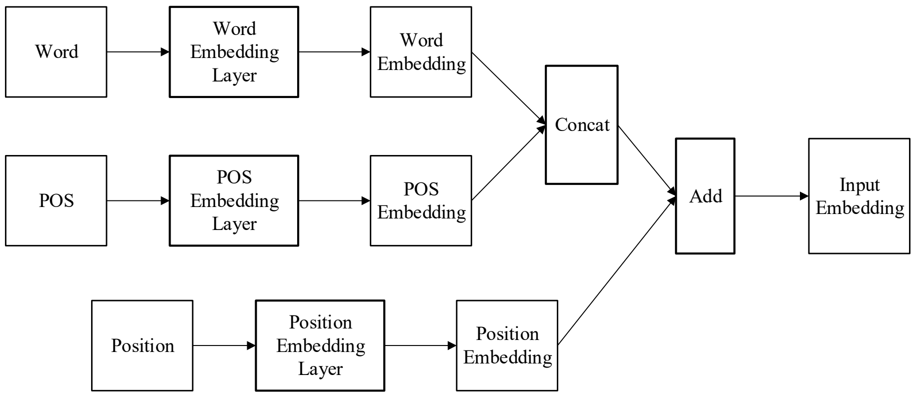

Figure 3. The input layer.

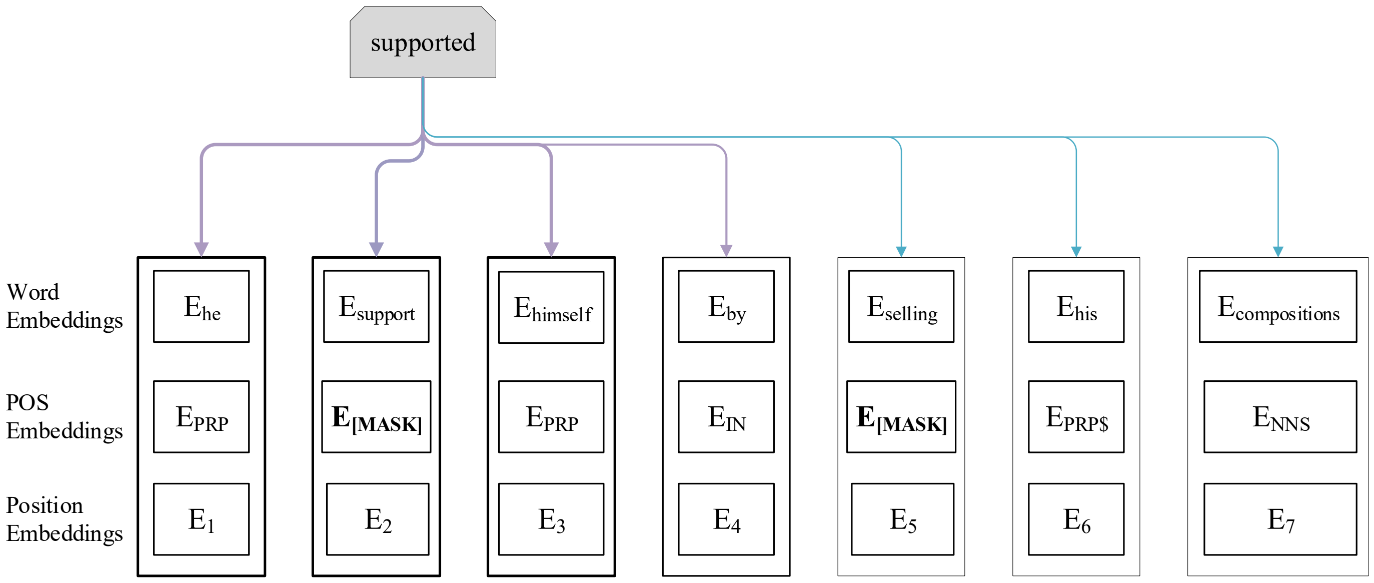

Figure 4. The self-attention mechanism in the model. The self-attention focuses not only on the word embedding and the position embedding of the token *supported* itself, but also on the information of the surrounding tokens, when the model predicts the POS tag of the token *supported*.

As a result of the composition of the input embedding, the self-attention mechanism in the encoder allows the model to attend to information from word embeddings, POS embeddings, and position embeddings, which satisfies the fact that POS tagging for a word depends not only the word itself but also on its position, its surrounding words, and their POS tags.

Figure 5 shows the structure of the model, which aims to predict the masked POS tags. The model takes tokens and POS tags as input, and they are transformed into input embeddings in the input layer. The Transformer's encoder is able to exploit bidirectional contextual information thanks to the masking and the self-attention mechanism. After the computation of the encoder, a linear layer with softmax function is used to compute the probability of each POS tag. Here, the model predicts POS tags of the token$_2$ and the token$_6$.

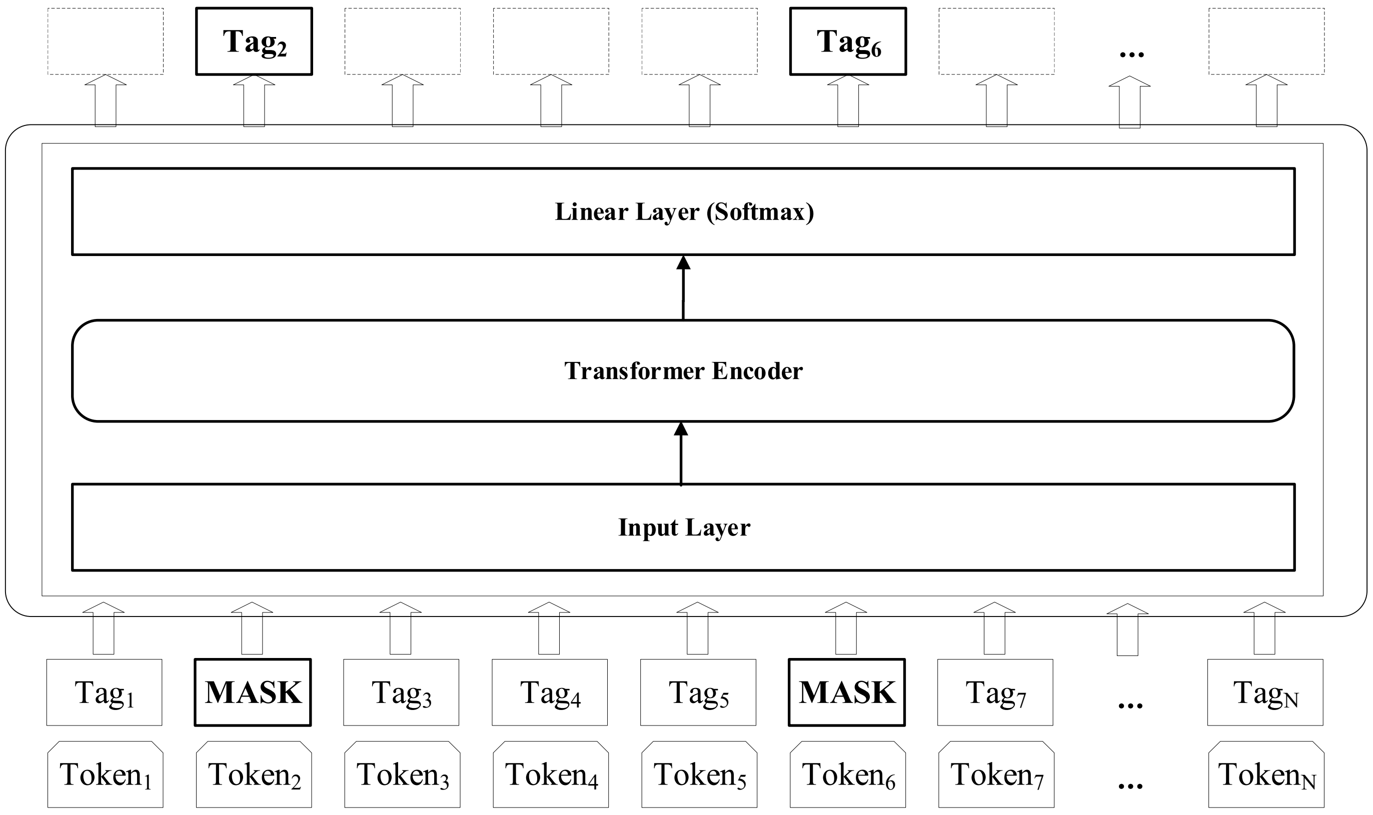

Figure 5. The structure of the model.

4.2. Training

We construct $\hat{y}$, which represents these POS tags that are not masked, and the training objective is to maximize the likelihood on training data

$$\underset{\theta}{max}\ log\ p_\theta(\overline{y}|w_1, w_2, \ldots, w_N, \hat{y}) \approx \sum_{i=1}^{N} m_i\ log\ p_\theta(y_i|w_1, w_2, \ldots w_N, \hat{y}) \tag{2}$$

where $\overline{y}$ represents masked POS tags.

4.3. Inference

For tokens of a given sentence, most are correctly tagged during the data preprocessing and there is no need to use deep learning models to predict their POS tags. If the candidate POS tagset of the token w_i contains more than one POS tag, its POS tag is predicted by the model. The most likely POS tag of the token w_i can be chosen as

$$y_i{}' = arg \underset{t \in 1,\ldots,k}{max} P_i(t|w_1, w_2, \ldots, w_N, \hat{y}) \tag{3}$$

where k is the number of tag types and $\hat{y}$ represents POS tags of tokens tagged by the data preprocessing.

5. Experiments

5.1. Dataset

Due to lack of the Penn Treebank WSJ dataset, we use the Groningen Meaning Bank (GMB) dataset [38], which is a large semantic annotated corpus. The dataset contains the annotation of all tokens with various tags, in which we only use the POS tags.

In the dataset, there are 62,010 sentences (1,354,149 tokens). After being shuffled, 80% of the dataset is split into the training set and 20% is split into the test set for validation. After the data preprocessing, POS tags of about 68% tokens are obtained. Therefore, POS tags of about 32% tokens are masked so as to be predicted by the model.

5.2. Settings

To configure the model for training, the optimizer used is Adam [39] with a learning rate of 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The loss function is set to cross entropy loss, and only masked POS tags participate in the computation of the loss function.

For the word embedding layer, it is initialized with 100-dimensional Glove word embeddings [19].

For the POS embedding layer, the dimensionality of POS embeddings is 10, which is enough to represent 48 Penn Treebank POS tags.

For the position embedding layer, absolute position embeddings are employed to encode the position, and the dimensionality is 110. Since the length of most sentences in the data source is less than 64, the max position is set to 64. If the length of a sentence is greater than 64, it will be truncated. Otherwise, it will be padded.

In the Transformer's encoder, two identical layers are stacked. For the multi-head attention sublayer, the number of attention heads is 11. For the feed-forward sublayer, the dimensionality of input and output is 110, and the dimensionality of the inner layer is 3072.

For the linear layer, the softmax function for multiclass classification is used, and its hidden size is 48.

To train the model, the PyTorch [40], an open source machine learning framework, is adopted. The version of the adopted PyTorch is 1.7.0. In the adopted PyTorch, the version of the CUDA toolkit is 10.1. In the following experiments, we use a batch size of 128 instances and train the model on RTX2060 for 100 epochs to report the results.

5.3. Evaluation

We evaluate our approach on two metrics: token accuracy and sentence accuracy. The token accuracy is the tag accuracy of per-token, and the sentence accuracy refers to the whole-sentence correct rate. The sentence accuracy is generally lower than the token accuracy, because a sentence is considered to be correctly labeled only if all the tokens of the sentence are correctly tagged.

The accuracy is jointly determined by the rule-based data preprocessing and the model. As shown in Figure 6, the token accuracy is more than 96% after one epoch because the data preprocessing has tagged most tokens (about 68%) before training. During the training of the model, both the token accuracy and the sentence accuracy are gradually improved. After 19 epochs, they reach a maximum on the test set. Specifically, the token accuracy increases to 98.60% and the sentence accuracy rises to 76.04%.

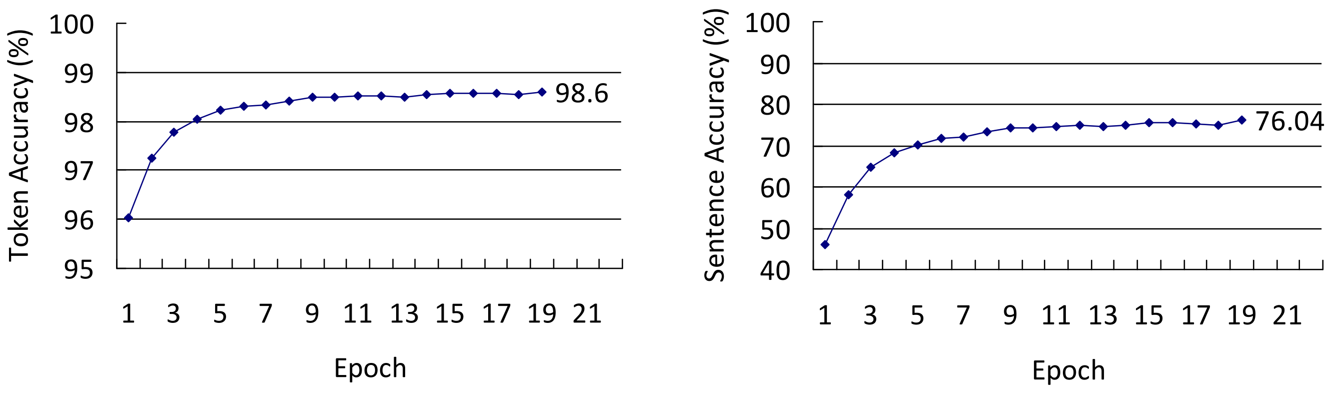

Figure 6. (**a**) The token accuracy of test set; (**b**) The sentence accuracy of test set.

For comparison with other methods, four models are chosen as baseline systems as follows.

- Bi-LSTM: A two-layer Bi-LSTM with the hidden size 50 is used, where we do not load pretrained word embeddings in word embedding layer.
- BLSTM RNN with word embedding [20]: In addition to a two-layer Bi-LSTM with the hidden size 100, a function is introduced to indicate original case of words. For a fair comparison, 100-dimensional Glove word embeddings are adopted in word embedding layer.
- C2W: A C2W model [21] is employed to generate 100-dimensional character-level embeddings of words, and a two-layer Bi-LSTM with the hidden size 100 takes the embeddings as input for POS tagging. The C2W model is composed of a character embedding layer and a unidirectional LSTM with the hidden size 100. For the character embedding layer in the C2W model, it generates 50-dimensional embeddings of characters, which are fed into the unidirectional LSTM to produce 100-dimensional character-level embeddings of words.
- Highway Bi-LSTM: A two-layer highway Bi-LSTM [23] with the hidden size 150 is adopted. The input to the highway Bi-LSTM comes from two parts: 100-dimensional Glove word embeddings and 50-dimensional character-level embeddings generated by a C2W model. In the C2W model, the character embedding layer yields 10-dimensional embeddings of characters and the unidirectional LSTM produces 50-dimensional character-level embeddings of words.

To verify the impact of each component of our method, two baselines are constructed as follows.

- Transformer's Encoder: The rule-based data preprocessing is removed from our method to verify whether the data preprocessing can deliver the performance gains. Without the rule-based data preprocessing, we cannot mask a certain portion of POS tags. Thus, the encoder of the Transformer is used to predict POS tags of all tokens.
- MLP with the data preprocessing: To verify the effectiveness of the self-attention mechanism on POS tagging, the multi-head attention layers are removed from the Transformer's encoder, which degenerates to a multilayer perceptron (MLP). With the rule-based data preprocessing, the MLP only predicts the masked POS tags. It is difficult to capture dependencies between words due to lack of the self-attention layers.

Table 4 shows the results of different methods on GMB dataset. We can see that our method outperforms all baselines in both the token accuracy and the sentence accuracy. The method achieves a token accuracy of 98.6% and a sentence accuracy of 76.04% on the dataset.

Table 4. Test set accuracy of different methods.

Model	Token Accuracy (%)	Sentence Accuracy (%)
Bi-LSTM	96.75	57.82
BLSTM RNN with word embedding	97.58	62.71
C2W	98.08	68.04
Highway Bi-LSTM	98.44	73.19
Transformer's Encoder	97.18	57.67
MLP with the data preprocessing	96.04	45.65
Ours	**98.60**	**76.04**

By observing the results of the Transformer's Encoder and the MLP with the data preprocessing, we can find that both the token accuracy and the sentence accuracy will drop if one of the components is removed from our method. This shows that all components of our method are indispensable. For other baselines, the C2W model performs better than the BLSTM RNN with word embedding, perhaps because the character-level embeddings contain richer semantic features than the word-level embeddings. In all baseline systems, the highway Bi-LSTM performs best, which can be attributed to the highway network and the richer input information.

Compared with the highway Bi-LSTM, our method without using the character-level embeddings achieves more competitive results. Specifically, our method boosts the sentence accuracy by about 3%. It is the combination of the rule-based data preprocessing and the deep learning model based on the self-attention that further improves the accuracy.

The above observations indicate that the rule-based data preprocessing is useful to improve the accuracy, and the self-attention mechanism can bring performance improvement by attending to information from word embeddings, position embeddings, and POS embeddings between words. The results demonstrate the effectiveness of our method.

6. Conclusions

In this paper, we propose a novel approach for POS tagging, including rule-based data preprocessing and a deep learning model based on Transformer. During the rule-based data preprocessing, most of the tokens are tagged, which enables the model to utilize their POS tags to predict the POS tags of the remaining tokens. By masking a certain portion of POS tags and utilizing the self-attention, the model is able to leverage bidirectional contexts. Our approach combines the rule-based methods with deep learning, which is helpful for the research on POS tagging. Experiments on the GMB dataset for POS tagging validate the effectiveness of the proposed method, and the proposed method achieves a per-token tag accuracy of 98.6% and a whole-sentence correct rate of 76.04% on the dataset.

7. Future Works

In the future, we plan to extend our approach to the research of POS tagging in other languages. Meanwhile, we can improve the rule-based data preprocessing and the deep learning model to further improve the accuracy of POS tagging.

Existing works on rule-based methods focus on how to correctly tag all words of a sentence, which is almost impossible due to the complexity of natural languages. In this paper, we provide a simple implementation of tagging some portion of the words. It is feasible and can improve the accuracy of POS tagging when combined with deep learning. However, the rules in this study only consider the local context and, thus, cannot satisfy all cases. There is still room to optimize in the construction of rules.

For the deep learning model, the Transformer is slightly insensitive to the position information of words, leading to insignificant improvement in the token accuracy. The absolute position embeddings can be replaced with relative position embeddings [41] to enhance the performance of the Transformer. Additionally, ELMo [42] can be employed to acquire the contextualized word representations in the input layer, which may allow the Transformer to make better use of contextual information.

We believe that our study is beneficial to POS tagging for the languages in which existing POS taggers perform poorly. Our approach can be applied to these languages by constructing rules corresponding to these languages in the data preprocessing and is excepted to improve the accuracy.

Author Contributions: Conceptualization, H.L. and H.M.; methodology, H.L., H.M. and J.W.; model, H.L.; validation, H.L., H.M. and J.W.; writing—original draft preparation, H.L.; writing—review and editing, H.L., H.M. and J.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (No. 61872145, 62032024), MOE International Joint Lab of Trustworthy Software, East China Normal University (No. 2021-5) and the Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University (No. ESSCKF2021-03).

Data Availability Statement: The GMB dataset [38] can be accessed at https://gmb.let.rug.nl/data.php (accessed on 12 October 2021).

Conflicts of Interest: The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

1. Hajic, J.; Ciaramita, M.; Johansson, R.; Kawahara, D.; Martí, M.A.; Màrquez, L.; Meyers, A.; Nivre, J.; Padó, S.; Stepánek, J.; et al. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009), Boulder, CO, USA, 4–5 June 2009; pp. 1–18.
2. Yang, X.; Liu, Y.; Xie, D.; Wang, X.; Balasubramanian, N. Latent part-of-speech sequences for neural machine translation. *arXiv* **2019**, arXiv:1908.11782.
3. Tan, Y.; Wang, X.; Jia, T. From syntactic structure to semantic relationship: Hypernym extraction from definitions by recurrent neural networks using the part of speech information. In Proceedings of the 19th International Semantic Web Conference, Athens, Greece, 2–6 November 2020.
4. Qi, P.; Zhang, Y.; Zhang, Y.; Bolton, J.; Manning, C.D. Stanza: A python natural language processing toolkit for many human languages. *arXiv* **2020**, arXiv:2003.07082.
5. Zhou, H.; Zhang, Y.; Li, Z.; Zhang, M. Is POS tagging necessary or even helpful for neural dependency parsing? In Proceedings of the Natural Language Processing and Chinese Computing, Zhengzhou, China, 14–18 October 2020; pp. 179–191.
6. Manning, C.D. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In Proceedings of the Computational Linguistics and Intelligent Text Processing, Tokyo, Japan, 20–26 February 2011; pp. 171–189.
7. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
8. Schuster, M.; Paliwal, K.K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing* **1997**, *45*, 2673–2681. [CrossRef]
9. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
10. Marcus, M.P.; Santorini, B.; Marcinkiewicz, M.A. Building a large annotated corpus of English: The penn treebank. *Comput. Linguist.* **1993**, *19*, 313–330.
11. Brill, E. A Simple rule-based part of speech tagger. In Proceedings of the Third Conference on Applied Natural Language Processing, Trento, Italy, 31 March–3 April 1992; pp. 152–155.
12. Brill, E. Some advances in transformation-based part of speech tagging. In Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, 31 July–4 August 1994; pp. 722–727.
13. Pandian, S.L.; Geetha, T.V. CRF models for tamil part of speech tagging and chunking. In Proceedings of the Computer Processing of Oriental Languages. Language Technology for the Knowledge-Based Economy, Hong Kong, 26–27 March 2009; pp. 11–22.
14. Albared, M.; Omar, N.; Aziz, M.J.A.; Ahmad Nazri, M.Z. Automatic part of speech tagging for arabic: An experiment using bigram hidden Markov model. In Proceedings of the Rough Set and Knowledge Technology, Beijing, China, 15–17 October 2010; pp. 361–370.
15. Horsmann, T.; Zesch, T. Do LSTMs really work so well for PoS tagging?—A replication study. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 7–11 September 2017.
16. Ma, X.; Hovy, E.H. End-to-end sequence labeling via Bi-directional LSTM-CNNs-CRF. *arXiv* **2016**, arXiv:1603.01354v5.
17. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the 27th Annual Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–8 December 2013.
18. Joulin, A.; Grave, E.; Bojanowski, P.; Mikolov, T. Bag of tricks for efficient text classification. *arXiv* **2016**, arXiv:1607.01759.

19. Pennington, J.; Socher, R.; Manning, C. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
20. Wang, P.; Qian, Y.; Soong, F.K.; He, L.; Zhao, H. Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network. *arXiv* **2015**, arXiv:1510.06168.
21. Ling, W.; Luís, T.; Marujo, L.; Astudillo, R.F.; Amir, S.; Dyer, C.; Black, A.W.; Trancoso, I. Finding function in form: Compositional character models for open vocabulary Word Representation. *arXiv* **2015**, arXiv:1508.02096.
22. Plank, B.; Søgaard, A.; Goldberg, Y. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv* **2016**, arXiv:1604.05529.
23. Srivastava, R.K.; Greff, K.; Schmidhuber, J. Highway networks. *arXiv* **2015**, arXiv:1505.00387.
24. Qi, P.; Dozat, T.; Zhang, Y.; Manning, C.D. Universal dependency parsing from scratch. *arXiv* **2019**, arXiv:1901.10457.
25. Dozat, T.; Qi, P.; Manning, C. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Vancouver, BC, Canada, 3–4 August 2017; pp. 20–30.
26. Warjri, S.; Pakray, P.; Lyngdoh, S.A.; Maji, A.K. Part-of-speech (POS) tagging using conditional random field (CRF) model for Khasi corpora. *Int. J. Speech Technol.* **2021**, *24*, 853–864. [CrossRef]
27. AlKhwiter, W.; Al-Twairesh, N. Part-of-speech tagging for Arabic tweets using CRF and Bi-LSTM. *Comput. Lang.* **2021**, *65*, 101138. [CrossRef]
28. Maimaiti, M.; Wumaier, A.; Abiderexiti, K.; Yibulayin, T. Bidirectional long short-term memory network with a conditional random field layer for uyghur part-of-speech tagging. *Information* **2017**, *8*, 157. [CrossRef]
29. Li, Z.; Sun, Y.; Tang, S.; Zhang, C.; Ma, H. Sentence-level semantic features guided adversarial network for zhuang language part-of-speech tagging. In Proceedings of the 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), Portland, OR, USA, 4–6 November 2019; pp. 265–272.
30. Gui, T.; Huang, H.; Peng, M.; Huang, X. Part-of-speech tagging for Twitter with adversarial neural networks. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 7–11 September 2017; pp. 2411–2420.
31. Yang, Z.; Salakhutdinov, R.; Cohen, W.W. Transfer Learning for Sequence Tagging with Hierarchical Recurrent Networks. *arXiv* **2017**, arXiv:1703.06345.
32. Wang, H.; Yang, J.; Zhang, Y. From genesis to creole language: Transfer learning for singlish universal dependencies parsing and POS tagging. *ACM Trans. Asian Low-Resource Lang. Inf. Processing* **2019**, *19*, 1–29. [CrossRef]
33. Kim, J.-K.; Kim, Y.-B.; Sarikaya, R.; Fosler-Lussier, E. Cross-lingual transfer learning for POS tagging without cross-lingual resources. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 7–11 September 2017; pp. 2832–2838.
34. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16×16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
35. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
36. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv* **2019**, arXiv:1907.11692.
37. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.G.; Salakhutdinov, R.; Le, Q.V. XLNet: Generalized autoregressive pretraining for language understanding. In Proceedings of the 32nd Annual Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, BC, Canada, 8–14 December 2019.
38. Bos, J.; Basile, V.; Evang, K.; Venhuizen, N.J.; Bjerva, J. The groningen meaning bank. In *Handbook of Linguistic Annotation*; Ide, N., Pustejovsky, J., Eds.; Springer: Dordrecht, The Netherlands, 2017; pp. 463–496.
39. Kingma, D.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
40. PyTorch. Available online: https://pytorch.org/ (accessed on 12 October 2021).
41. Huang, Z.; Liang, D.; Xu, P.; Xiang, B. Improve transformer models with better relative position embeddings. *arXiv* **2020**, arXiv:2009.13658.
42. Peters, M.E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep contextualized word representations. *arXiv* **2018**, arXiv:1802.05365.