N-LTP: An Open-source Neural Language Technology Platform for Chinese

Wanxiang Che, Yunlong Feng, Libo Qin, Ting Liu
Research Center for Social Computing and Information Retrieval
Harbin Institute of Technology, China
{car,ylfeng,lbqin,tliu}@ir.hit.edu.cn

Abstract

We introduce N-LTP, an open-source neural language technology platform supporting six fundamental Chinese NLP tasks: lexical analysis (Chinese word segmentation, part-of-speech tagging, and named entity recognition), syntactic parsing (dependency parsing), and semantic parsing (semantic dependency parsing and semantic role labeling). Unlike the existing state-of-the-art toolkits, such as Stanza, that adopt an independent model for each task, N-LTP adopts the multi-task framework by using a shared pre-trained model, which has the advantage of capturing the shared knowledge across relevant Chinese tasks. In addition, a knowledge distillation method (Clark et al., 2019) where the single-task model teaches the multi-task model is further introduced to encourage the multi-task model to surpass its single-task teacher. Finally, we provide a collection of easy-to-use APIs and a visualization tool to make users to use and view the processing results more easily and directly. To the best of our knowledge, this is the first toolkit to support six Chinese NLP fundamental tasks. Source code, documentation, and pre-trained models are available at https://github.com/HIT-SCIR/ltp.

1 Introduction

There is a wide range of existing natural language processing (NLP) toolkits such as CoreNLP (Manning et al., 2014), UDPipe (Straka and Straková, 2017), FLAIR (Akbik et al., 2019), spaCy, and Stanza (Qi et al., 2020) in English, which makes it easier for users to build tools with sophisticated linguistic processing. Recently, the need for Chinese NLP has a dramatic increase in many downstream applications. A Chinese NLP platform usually includes lexical analysis (Chinese word segmentation (CWS), part-of-speech (POS) tagging, and named entity recognition (NER)), syntactic parsing (dependency parsing (DEP)), and semantic parsing (semantic dependency parsing (SDP) and semantic role labeling (SRL)). Unfortunately, there are relatively fewer high-performance and high-efficiency toolkits for Chinese NLP tasks. To fill this gap, it’s important to build a Chinese NLP toolkit to support rich Chinese fundamental NLP tasks, and make researchers process NLP tasks in Chinese quickly.

Recently, Qi et al. (2020) introduce the Python NLP toolkit Stanza for multi-lingual languages, including Chinese language. Though Stanza can be directly applied for processing the Chinese texts, it suffers from several limitations. First, it only supports part of Chinese NLP tasks. For example, it fails to handle semantic parsing analysis, resulting in incomplete analysis in Chinese NLP. Second, it trained each task separately, ignoring the shared knowledge across the related tasks, which has been proven effective for Chinese NLP tasks (Qian et al., 2015; Hsieh et al., 2017; Chang et al., 2018). Third, independent modeling method will occupy more...
Table 1: Feature comparisons of N-LTP against other popular natural language processing toolkits.

<table>
<thead>
<tr>
<th>System</th>
<th>Programming Language</th>
<th>Fully Neural</th>
<th>State-of-the-art Performance</th>
<th>Rich Chinese Fundamental Tasks</th>
<th>Multi-task Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTP (Che et al., 2010)</td>
<td>C++</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>UDPipe (Straka and Straková, 2017)</td>
<td>C++</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FLAIR (Akbik et al., 2019)</td>
<td>Python</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Stanza (Qi et al., 2020)</td>
<td>Python</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>N-LTP</td>
<td>Python</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

memory with the increase of the number of tasks, which makes it hard to deploy for mobile devices in real-world scenario.

To address the aforementioned issues, we introduce N-LTP, a PyTorch-based neural natural language processing toolkit for Chinese NLP, which was built on the SOTA pre-trained model. As shown in Figure 1, given Chinese corpus as input, N-LTP produces comprehensive analysis results, including lexical analysis, syntactic parsing, and semantic parsing. In addition, N-LTP provides easy-to-use APIs and visualization tool, which is user-friendly.

As shown in Table 1, compared to the existing widely-used NLP toolkits, N-LTP has the following advantages:

- **Comprehensive Tasks.** N-LTP supports rich Chinese fundamental NLP tasks including lexical analysis (word segmentation, part-of-speech tagging, named entity recognition), syntactic parsing, and semantic parsing (semantic dependency parsing, semantic role labeling). To the best of our knowledge, this is the first neural Chinese toolkit that support six Chinese fundamental NLP tasks.

- **Multi-Task Learning.** The existing NLP toolkits for the Chinese language all adopt independent models for each task, which ignore the shared knowledge across tasks.

To alleviate this issue, we propose to use the multi-task framework (Collobert et al., 2011) to take advantage of the shared knowledge across all tasks. Meanwhile, multi-task learning with a shared encoder for all six tasks can greatly reduce the occupied memory and improve the speed, which makes N-LTP more efficient, reducing the need for hardware.

In addition, to enable the multi-task learning to enhance each subtask performance, we follow Clark et al. (2019) to adopt the distillation method single-task models teach a multi-task model, helping the multi-task model surpass its all single-task teachers.

- **Extensibility.** N-LTP works with users’ custom modules. Users can easily add a new pre-trained model with a configuration file, in which users can change the pretrained model to any BERT-like model supported by HuggingFace Transformers (Wolf et al., 2019) easily by changing the config. We have made all task training configuration files open-sourced.

- **Easy-to-use API and Visualization Tool.** N-LTP provides a collection of fundamental APIs, which is convenient for users to use the toolkit without the need for any knowledge. We also provide a visualization tool, which enables users to view the processing results directly. In addition, N-LTP has bindings for many programming languages (C++, Python, Java, Rust, etc.).

- **State-of-the-art Performance.** We evaluate N-LTP on a total of six Chinese NLP tasks, and find that it achieves state-of-the-art or competitive performance at each task.

N-LTP is fully open-sourced and can support six Chinese fundamental NLP tasks. We hope N-LTP can facilitate Chinese NLP research.

2 Design and Architecture

Figure 2 shows an overview of the main architecture of N-LTP. It mainly consists of the components including a shared encoder and different decoders for each task. Our framework shares one encoder for leveraging the shared knowledge across all tasks. Different task decoders are used for each task separately. All tasks are optimized simultaneously via a joint learning scheme. In addition, the knowledge distillation technique is introduced to encourage the multi-task model to surpass its single-task teacher model.
2.1 Shared Encoder

Multi-task framework uses a shared encoder to extract the shared knowledge across related tasks, which has obtained remarkable success on various NLP tasks (Qin et al., 2019; Wang et al., 2020; Zhou et al., 2021). Inspired by this, we adopt the SOTA pre-trained model (ELECTRA) (Clark et al., 2020) as the shared encoder to capture shared knowledge across six Chinese tasks.

Given an input utterance \( s = (s_1, s_2, \ldots, s_n) \), we first construct the input sequence by adding specific tokens \( s = ([\text{CLS}], s_1, s_2, \ldots, s_n, [\text{SEP}]) \), where \([\text{CLS}]\) is the special symbol for representing the whole sequence, and \([\text{SEP}]\) is the special symbol to separate non-consecutive token sequences (Devlin et al., 2019). ELECTRA takes the constructed input and output the corresponding hidden representations of sequence \( H = (h_{[\text{CLS}]}, h_1, h_2, \ldots, h_n, h_{[\text{SEP}]}) \).

2.2 Chinese Word Segmentation

Chinese word segmentation (CWS) is a preliminary and important task for Chinese natural language processing (NLP). In N–LTP, following Xue (2003), CWS is regarded as a character based sequence labeling problem.

Specifically, given the hidden representations \( H = (h_{[\text{CLS}]}, h_1, h_2, \ldots, h_n, h_{[\text{SEP}]}) \), we adopt a linear decoder to classify each character:

\[
y_i = \text{Softmax}(W_{\text{CWS}} h_i + b_{\text{CWS}}),
\]

where \( y_i \) denotes the label probability distribution of each character; \( W_{\text{CWS}} \) and \( b_{\text{CWS}} \) are trainable parameters.

2.3 POS Tagging

Part-of-speech (POS) tagging is another fundamental NLP task, which can facilitate the downstream tasks such as syntactic parsing. Following the dominant model in the literature (Ratnaparkhi, 1996; Huang et al., 2015), POS tagging can be treated as a sequence labeling task.

Similar to CWS, we take the sequence of hidden representations \( H \) as input and output the corresponding POS sequence labels, which is formulated as:

\[
y_i = \text{Softmax}(W_{\text{POS}} h_i + b_{\text{POS}}),
\]

where \( y_i \) denotes the POS label probability distribution of the \( i \)-th character; \( h_i \) is the first sub-token representation of word \( s_i \).

2.4 Named Entity Recognition

The named entity recognition (NER) is the task of finding the start and end of an entity (people, locations, organizations, etc.) in a sentence and assigning a class for this entity.

Traditional, NER is regarded as a sequence labeling task. After obtaining the hidden representations \( H \), we follow Yan et al. (2019a) to adopt the Adapted-Transformer to consider direction- and distance-aware characteristic, which can be formulated as:

\[
\hat{h}_i = \text{AdaptedTransformer}(h_i),
\]

where \( \hat{H} = (\hat{h}_{[\text{CLS]}}, \hat{h}_1, \hat{h}_2, \ldots, \hat{h}_n, \hat{h}_{[\text{SEP}]} \) are the updated representations.

Finally, similar to CWS and POS, we use a linear decoder to classify label for each word:

\[
y_i = \text{Softmax}(W_{\text{NER}} \hat{h}_i + b_{\text{NER}}),
\]

where \( y_i \) denotes the NER label probability distribution of each character.

2.5 Dependency Parsing

Dependency parsing is the task to analyze the semantic structure of a sentence. In N–LTP, we implement a deep biaffine neural dependency parser (Dozat and Manning, 2017) and einser algorithm (Eisner, 1996) to obtain the parsing result, which is formulated as:

\[
\begin{align*}
r_i^{(\text{head})} &= \text{MLP}^{(\text{head})}(h_i) \\
r_j^{(\text{dep})} &= \text{MLP}^{(\text{dep})}(h_j)
\end{align*}
\]

After obtaining \( r_i^{(\text{head})} \) and \( r_j^{(\text{dep})} \), we compute the score for each dependency \( i \sim j \) by:

\[
y_{i \sim j} = \text{BiAffine}(r_i^{\text{dep}}, r_j^{\text{head}}).
\]
The above process is also used for scoring a labeled dependency $i \overset{\lambda}{\leftarrow} j$, by extending the 1-dim vector $s$ into $L$ dims, where $L$ is the total number of dependency labels.

### 2.6 Semantic Dependency Parsing

Similar to dependency parsing, semantic dependency parsing (Che et al., 2012, SDP) is a task to capture the semantic structure of a sentence. Specifically, given an input sentence, SDP aims at determining all the word pairs related to each other semantically and assigning specific predefined semantic relations. Following Dozat and Manning (2017), we adopt a biaffine module to perform the task, using

$$ p_{i \overset{\lambda}{\leftarrow} j} = \text{sigmoid}(y_{i \overset{\lambda}{\leftarrow} j}). \quad (7) $$

If $p_{i \overset{\lambda}{\leftarrow} j} > 0.5$, word $i$ to word $j$ exists an edge.

### 2.7 Semantic Role Labeling

Semantic Role Labeling (SRL) is the task of determining the latent predicate-argument structure of a sentence, which can provide representations to answer basic questions about sentence meaning, including who did what to whom, etc. We adopt an end-to-end SRL model by combining a deep biaffine neural network and a conditional random field (CRF)-based decoder (Cai et al., 2018).

The biaffine module is similar to Section 2.5 and the CRF layer can be formulated as:

$$ P(\hat{y}|s) = \frac{\sum_{j=1}^{y} \exp f(y_{i,j-1}, y_{i,j}, s)}{\sum_{j=1}^{y} \exp f(y_{i,j-1}, y_{i,j}, s)} \quad (8) $$

where $\hat{y}$ represents an arbitrary label sequence when predicate is $s$, and $f(y_{i,j-1}, y_{i,j}, s)$ computes the transition score from $y_{i,j-1}$ to $y_{i,j}$.

### 2.8 Knowledge Distillation

When there exist a large number of tasks, it’s difficult to ensure that each task task benefits from multi-task learning (Clark et al., 2019).

Therefore, we follow BAM (Clark et al., 2019) to use the knowledge distillation to alleviate this issue, which is shown Figure 3. First, we train each task as the teacher model. Then, N–LTP learns from each trained single-task teacher model while learning from the gold-standard labels simultaneously.

Following BAM (Clark et al., 2019), we adopt teacher annealing distillation algorithm. More specifically, instead of simply shuffling the datasets for our multi-task models, we follow the task sampling procedure from Bowman et al. (2018), where the probability of training on an example for a particular task $\tau$ is proportional to $|D_\tau|^{0.75}$. This ensures that tasks with large datasets don’t overly dominate the training.

## 3 Usage

N–LTP is a PyTorch-based Chinese NLP toolkit based on the above model. All the configurations can be initialized from JSON files, and thus it is easy for users to use N–LTP where users just need one line of code to load the model or process the input sentences. Specifically, N–LTP can be installed easily by the command:

```bash
$ pip install ltp
```

In addition, N–LTP has bindings available for many programming languages, including C++, Python, Java and RUST directly.

### 3.1 Easy-to-use API

We provide rich easy-to-use APIs, which enables users to easily use without the need for any knowledge. The following code snippet in Figure 4 shows

```python
from ltp import LTP
ltp = LTP()
seg, hidden = ltp.seg(\"他叫汤姆去拿外衣。\")
pos = ltp.pos(hidden)
nl = ltp.nl(hidden)
sdp = ltp.sdp(hidden)
```
### Table 2: Main Results

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Stanza (Qi et al., 2020)</td>
<td>92.40</td>
<td>98.10</td>
<td>89.50</td>
<td>84.98</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>N-LTP trained separately</td>
<td>98.55</td>
<td>95.41</td>
<td>90.12</td>
<td>74.47</td>
<td>79.23</td>
<td></td>
</tr>
<tr>
<td>N-LTP trained jointly with distillation</td>
<td>99.18</td>
<td>95.97</td>
<td>90.19</td>
<td>76.62</td>
<td>79.49</td>
<td></td>
</tr>
</tbody>
</table>

“-” represents the absence of tasks in the Stanza toolkit and we cannot report the results.

The N-LTP model is based on the Chinese ELECTRA base (Cui et al., 2020). The learning ratio (lr) for teacher models, student model and CRF layer is \(\{1e^{-4}\}, \{1e^{-4}\}, \{1e^{-3}\}\), respectively. The gradient clip value adopted in our experiment is 1.0 and the warmup proportion is 0.02. We use BertAdam (Devlin et al., 2019) to optimize the parameters and adopted the suggested hyper-parameters for optimization.

### 4.2 Results

We compare N-LTP with the state-of-the-art toolkit Stanza. For a fair comparison, we conduct experiments on the same datasets that Stanza adopted.

The results are shown in Table 2, we have the following observations:

- N-LTP outperforms Stanza on four common tasks including CWS, POS, NER, and DEP by a large margin, which shows the superiority of our proposed toolkit.
- The multi-task learning outperforms the model with independently trained. This is because that the multi-task framework can consider the shared knowledge which can promote each task compared with the independently training paradigm.

### 4.3 Analysis

#### 4.3.1 Speedup and Memory Reduction

In this section, we perform the speed and memory test on the Tesla V100-SXM2-16GB and all models were speed-tested on the 10,000 sentences of the People’s Daily corpus with a batch size of 8. In all experiments, N-LTP performs six tasks (CWS, POS, NER, DEP, SDP, SRL) while Stanza only conduct four tasks (CWS, POS, NER, DEP).

Figure 5: LTP annotates a Chinese sentence “他叫汤姆去拿外衣.” / He told Tom to get his coat.”. The output is visualized by our visualization demo.

Figure 5: LTP annotates a Chinese sentence “他叫汤姆去拿外衣.” / He told Tom to get his coat.”. The output is visualized by our visualization demo.

### 3.2 Visualization Tool

In addition, a visualization tool is proposed for downloading models, annotating a sentence with customized models, and predicting all annotations.

#### 4.1 Experimental Setting

To evaluate the efficiency of our multi-task model, we conduct experiments on six Chinese tasks.
Task Model Dataset Metric State-of-the-art N-LTP N-LTP
CWS BERT (Huang et al., 2019) 10 Corpus2 F1 97.10 97.42 97.50
POS Glyce+BERT (Meng et al., 2019) CTB9 F1 93.15 94.57 95.17
NER ZEN (Diao et al., 2020) MSRA F1 95.25 94.95 95.78
NER DGLSTM-CRF (Jie and Lu, 2019) OntoNotes F1 79.92 84.08 84.38
NER Joint-Multi-BERT (Yan et al., 2019b) CONLL12 F1 81.71 81.69 84.03
SDP SuPar3 CCL20204 F1_LAS 80.38 76.27 75.76

Table 3: The results of N-LTP comparison to other state-of-the-art performance.

#### Speedup
We compare the speed between Stanza, N-LTP-separately and N-LTP-jointly and the results are shown in Figure 6. From the results of speed test, we have two interesting observations: (1) N-LTP trained separately achieves the x1.7 speedup compared with Stanza. We attribute that N-LTP adopts the transformer as an encoder that can be calculated in parallel while Stanza uses LSTM which can only process sentences word by word; (2) N-LTP trained jointly with distillation obtains the x4.3 speedup compared with separate modeling paradigm. This is because that our model utilizes the multi-task to perform all tasks while the independent models can be only processed all tasks in a pipeline mode.

#### Memory Reduction
For memory test, we have the following observation: (1) N-LTP trained separately occupy more memory than Stanza. This is because N-LTP performs six tasks while Stanza only conduct four tasks. (2) Though performing six tasks, N-LTP trained jointly only requires half the memory compared to Stanza. We attribute it to the fact that the multi-task framework with a shared encoder can greatly reduce the running memory.

#### 4.3.2 Comparison with Other SOTA Single Models
To further verify the effectiveness of N-LTP, we compare our framework with the existing state-of-the-art single models on six Chinese fundamental tasks. In this comparison, we conduct experiments on the same wildly-used dataset in each task for a fair comparison. In addition, we use BERT rather than ELECTRA as the shared encoder, because the prior work adopts BERT.

Table 3 shows the results, we observe that our framework obtains best performance on five out of six tasks including CWS, POS, NER, SRL, and DEP, which demonstrates the effectiveness of our framework. On the SDP task, N-LTP underperforms the best baseline. This is because many tricks are used in the prior model for SDP task and we just use the basic multi-task framework.

#### 5 Conclusion
In this paper, we presented N-LTP, an open-source neural language technology platform supporting Chinese. To the best of our knowledge, this is the first Chinese toolkit that supports six fundamental Chinese NLP tasks. Experimental results show N-LTP obtains state-of-the-art or competitive performance and has high speed. We hope N-LTP can facilitate Chinese NLP research.

#### Acknowledgements
We thank the anonymous reviewers for their detailed and constructive comments. The first three authors contributed equally. Wanxiang Che is the corresponding author. This work was supported by the National Key R&D Program of China via grant 2020AAA0106501 and the National Natural Science Foundation of China (NSFC) via grant 61976072 and 61772153. Libo is also supported by the Zhejiang Lab’s International Talent Fund for Young Professionals.
References


