Tibetan Dependency Parsing with Graph Convolutional Neural Networks

Bo An
Institute of Ethnology and Anthropology,Chinese Academy of Social Sciences
Building 6, Zhongguancun Nandajie 27, Beijing, China
anbo@cass.org.cn

Abstract

Dependency parsing is a syntactic analysis method to analyze the dependency relationships between words in a sentence. The interconnection between words through dependency relationships is typical graph data. Traditional Tibetan dependency parsing methods typically model dependency analysis as a transition-based or sequence-labeling task, ignoring the graph information between words. We propose a graph neural network (GNN)-based Tibetan dependency parsing method to address this issue. This method treats Tibetan words as nodes and the dependency relationships between words as edges, thereby constructing the graph data of Tibetan sentences. Specifically, we use BiLSTM to learn the word representations of Tibetan, utilize GNN to model the relationships between words, and employ MLP to predict the types of relationships between words. We conduct experiments on a Tibetan dependency database, and the results show that the proposed method can achieve high-quality Tibetan dependency parsing results.

1 Introduction

In recent years, the explosive growth of Tibetan text data, fueled by the popularization of information technology in Tibetan areas, has made the processing and deeper understanding of Tibetan information a hot research topic in Tibetan natural language processing (NLP) (Faggionato and Meelen, 2019). Dependency analysis is an essential task for the semantic modeling of texts, as it provides a basis for deep semantic analysis and has significant research and practical value. The results of dependency analysis can be directly applied to numerous basic natural language processing tasks, such as question answering (Cao et al., 2019), sentiment analysis (Xiaomei et al., 2018), and named entity recognition (Jie et al., 2017).

Traditional Tibetan dependency analysis methods can be mainly classified into two categories: (1) statistical learning-based methods (Hua et al., 2013), which usually require experts to design corresponding rules and features and then use statistical learning models to model and predict dependency syntax. This type of method heavily relies on Tibetan linguistic experts. (2) deep learning-based methods(An and Long, 2021) , which have been widely applied in Tibetan information processing, such as word segmentation, text classification, and dependency analysis, with the rapid development of deep learning. The major advantage of deep learning-based methods is that they do not require expert features. Tibetan dependency analysis can be achieved through a fixed network structure and annotated data.

However, the methods above model Tibetan dependency analysis as a classification or transition problem, ignoring the features of graph data in dependency analysis. Graph data features can better model the relationships between different words and ignore the distance between words in the text, i.e., they can model the dependency information between words that are far apart. They can also model higher-order relationships through indirect relationships between words, which significantly impacts modeling word relationships, such as AMR (Abstract Meaning Representation) (Wang et al., 2020).

This paper presents a method for Tibetan dependency analysis based on graph convolutional neural networks. Tibetan word representations are modeled using Bert (Devlin et al., 2018) and BiLSTM, followed by graph neural networks(GNN) (Zhou et al., 2020) for

Proceedings of the Ancient Language Processing Workshop associated with RANLP-2023, pages 213–221,
held in Varna Bulgaria, Sept 8, 2023.
https://doi.org/10.26615/978-954-452-087-8_024

modeling dependency relationships between words. MLP is then employed for relationship classification and determining the dependency relationship types. The results on Tibetan dependency analysis data indicate that GNN can significantly enhance the performance of Tibetan dependency analysis, thus affirming the value of graph information.

The main contributions of our work are as follows: (1) We propose using graph convolutional neural networks (GCN) to model the dependency relationships in Tibetan sentences. (2) Experimental results show that the proposed method outperforms other methods, such as R-GCN (Schlichtkrull et al., 2018), in Tibetan dependency analysis, which may be due to insufficient training data.

The main contributions are twofold:

- We propose using GCN to model the dependency relationships in Tibetan sentences.
- Experimental results show that the GCN+MLP method outperforms other methods, such as R-GCN, in Tibetan dependency analysis, which may be due to insufficient training data.

The rest of the paper is organized as follows: Section 2 introduces some of the most related work, including Tibetan dependency parsing models and graph neural networks. Our proposed model is detailed described in Section 3. Section 4 shows our experimental results on the introduced Tibetan dependency analysis dataset and presents the effects of different modules. We conclude our work in Section 5.

2 Related Work

This section briefly reviews related work, including Tibetan dependency parsing methods and neural-based methods for dependency parsing.

2.1 Tibetan dependency parsing method

The Tibetan dependency analysis dataset is the foundation for researching dependency analysis methods. Therefore, the existing Tibetan dependency analysis data is introduced first. The current Tibetan dependency analysis dataset includes the following: For instance,Hua et al. 2013 construct a Tibetan dependency tree semi-automatically. It includes a word-pairs dependency classification model, and dependency edges annotation model based on Tibetan language grammar. Tashi and Duo 2015 built a Tibetan dependency treebank of multidimensional windows based on their grammar. Toudan et al. 2018 annotated dependency trees for sentences from Tibetan primary school textbooks. Wu et al. 2019 introduced a Tibetan dependency analysis dataset with 1500 sentences annotated based on complex dependency grammar with 62 types of dependency arcs. (An and Long, 2021) constructs a Tibetan dependency parsing dataset with more than 5000 Tibetan sentences based on an interlinearized annotation dataset.

Currently, most of the Tibetan dependency parsing models are composed of two components: feature extraction and dependency prediction. A discriminant model is proposed to conduct Tibetan dependency parsing based on feature engineering by Tibetan experts (quecai rang and Zhao, 2013). And their model was further utilized for the parsing of Tibetan compound sentences. Xia et al. 2019 extracted unigram, bigram, trigram, and some Tibetan-specific features for each word in the sentence and employed a perceptron classifier to perform Tibetan dependency parsing. All of the above works were based on features designed by Tibetan language experts. Compared with these methods, the main advantage of our method is that our model can extract useful feature vectors automatically.

2.2 Neural-based method for dependency analysis

In recent years, neural-based models have achieved competitive performances in many natural language processing tasks, such as word segmentation, part-of-speech, and semantic parsing. Furthermore, this line of works have two advantages: avoiding complex feature engineering and better generalization. Due to the above advantages, neural-based models are introduced for dependency analysis. There are two main research directions for dependency analysis: translation-based parsers and graph-based parsers.

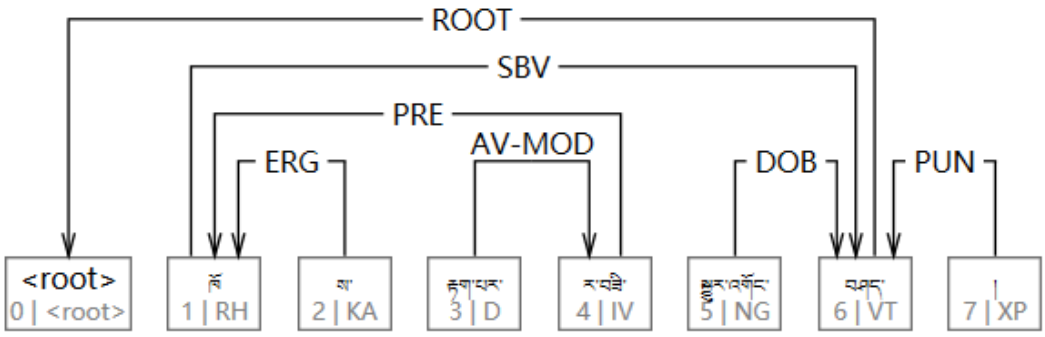

Figure 1: An example of Tibetan dependency tree.

Chen and Manning introduced the first neural translation-based dependency parser (Chen and Manning, 2014), which utilizes a feedforward network to assign a probability to each action the parser. Andor et al. (Andor et al., 2016) augments the above model with a beam search and a conditional random field loss objective for correcting false predictions. The Long-Short-Term Memory (LSTM) model was employed to achieve the state-of-the-art performance (Dyer et al., 2015; Kuncoro et al., 2016).

The first neural graph-based parser 2016 utilizes the attention mechanism from machine translation and LSTM to conduct dependency parsing. Hashimoto et al. 2016 extend the graph-based parser as a multi-task neural model and employ a bilinear MLP label classifier. Furthermore, Cheng et al. 2016 further resolve the limitation of being unable to condition the scores of each possible arc on previous parsing decisions of other graph-based parsers. Dozat et al. 2016 propose bi-affine classifiers to predict arcs and labels for dependency analysis tasks and achieve state-of-the-art performances. Recently, with the wide use of deep contextual embeddings (Peters et al., 2018), Schuster et al. 2019 introduced a multilingual transfer framework that utilizes deep contextual embeddings in an unsupervised fashion.

(An and Long, 2021) proposes a deep learning-based Tibetan dependency parsing method using BiLSTM and multi-layer perceptron. (Duo et al., 2021) models the Tibetan dependency parsing task using deep learning-based transition. Recently, scholars have introduced deep learning methods to the task of Tibetan dependency parsing task, resulting in an improvement in the performance of Tibetan dependency parsing. Despite the potential of graph neural networks, they have not been utilized in Tibetan dependency parsing tasks. Hence, this paper proposes a graph neural network-based Tibetan dependency parsing method to better model the relationships between words.

3 The GCN-based Tibetan Dependency Parsing Method

3.1 Task Definition

Dependency parsing is a natural language processing task that involves analyzing the grammatical structure of a sentence by identifying the relationships between the words in it. Moreover, Figure 1 presents an example of a Tibetan dependency tree.

Specifically, given a sentence S consisting of n words $w_1, w_2, ..., w_n$, dependency parsing aims to construct a directed acyclic graph $G = (V, E)$, where $V = v_1, v_2, ..., v_n$ is the set of vertices representing the words in S, and $E \subseteq V \times V$ is the set of directed edges representing the syntactic dependencies between words.

Each edge $e_{i,j} = (v_i, v_j)$ in E is labeled with a dependency type $r_{i,j} \in R$, where R is the set of all possible dependency types. The dependency tree's root is the vertex with no incoming edges. Thus, the dependency tree $T = (V, E')$ is a tree if it contains $n - 1$ edges and satisfies the constraints mentioned above.

The output of a dependency parser is the dependency tree T that represents the sentence's grammatical structure. This tree can be used for various downstream applications, such as machine translation, information retrieval, and text summarization.

This work employs the dataset introduced by (An and Long, 2021). There are 34 types of dependency arcs in our Tibetan dependency grammar. We present them in Table (1).

3.2 GNN for Tibetan Dependency Parsing

Tibetan dependency parsing includes word segmentation, word relation, and arc label prediction. The framework of Tibetan dependency parsing is presented in Figure 2.

We employ SegT (Huidan Liu and Yeping, 2012) for Tibetan word segmentation in this work. We utilize the Tibetan-Roberta-base to implement the embedding layer, which generates the embedding for each Tibetan syllable. Moreover, we employ BiLSTM (Kiperwasser and Goldberg, 2016) to compose the syllable embeddings into the word embedding h_i.

We employ Graph Convolutional Networks (GCN) to model Tibetan dependency parsing. GCNs can be used to model dependency parsing by constructing a graph representation of the sentence, where the vertices represent the words in the sentence, and the edges represent the syntactic dependencies between them. Each vertex is associated with a word embedding h_i, which captures the semantic information of the word. Formally, let $G = (V, E)$ be the graph representation of the sentence, where $V = v_1, v_2, ..., v_n$ is the set of vertices representing the words in the sentence, and $E \subseteq V \times V$ is the set of directed edges representing the syntactic dependencies between words. Each vertex v_i is associated with a word embedding $h_i \in \mathbb{R}^d$, where d is the dimension of the embedding. To capture the interactions between the vertices in the graph, GCNs perform graph convolution operations on the vertex embeddings. Specifically, the embedding of each vertex is updated by aggregating the embeddings of its neighboring vertices, weighted by an adjacency matrix A that encodes the edge information. The graph convolution operation can be expressed as:

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \frac{1}{c_{i,j}} W^{(l)} h_j^{(l)}\right)$$

where $h_i^{(l)}$ is the embedding of vertex i at layer l, $\mathcal{N}(i)$ is the set of neighboring vertices of vertex i, $W^{(l)}$ is the weight matrix at layer l, and $c_{i,j}$ is a normalization constant that ensures that the sum of the weights of the neighbors of vertex i is 1. The activation function σ is typically a non-linear function, such as the rectified linear unit (ReLU).

After several graph convolution operations, the final vertex embeddings can be fed into a classifier to predict the syntactic dependency labels between the words.

3.2.1 Arc Prediction Layer

This layer comprises two classifiers; the first predicts the dependency head for each word, while the second classifies the type of dependency arc between the word and its head word.

Head Classifier. The input to this classifier is the feature vectors of each word, and it outputs the index of the word's head. Since the number of words in a sentence is variable, this is a variable-class classification task, making it impossible to utilize a multi-layer perceptron (MLP) typically used for category tasks. We draw inspiration from the biaffine attention model (Dozat and Manning, 2016) to address this challenge and employ two MLP models to build the head classifier. Specifically, we use Equation (1) and (2) to convert the feature vector of each word into two vectors $\vec{f}^{head} \in d^h$ and $\vec{f}^{dep} \in d^h$, respectively, to represent the head and dependent nodes of the dependency arc.

$$\vec{f}_i^{head} = MLP^{head}\vec{f}_i \tag{1}$$

$$f_i^{dep} = MLP^{dep}\vec{f}_i \tag{2}$$

Next, the position of the head node for each word i is calculated using a bilinear attention mechanism as per Equation (3), where $h_i^{arc} \in d^h$ represents the score for the j-th word as the head node of the i-th word. Here, $\mathbf{F^{dep}} \in \mathbf{d^{nh}}$ is a matrix obtained by concatenating the head representation $\vec{f}_i^{head}$ of all words in the sentence, where n is the number of words in the sentence. Additionally, $U \in d^{hh}$ is the parameter matrix, and $\vec{u} \in d^h$ is the parameter vector.

$$h_j^{arc} = \mathbf{F^{dep}}\mathbf{U}\mathbf{f_i^{dep}} + \mathbf{F^{head}}\mathbf{u} \tag{3}$$

Table 1: The types of Tibetan dependency arcs.

Item	Dependency Relationship	Label	Item	Dependency Relationship	Label
1	Subject predicate relationship	SBV	2	Direct object relationship	DOB
3	Indirect object relationship	IOB	4	Subject verb relationship	SBC
5	Predicative verb relationship	CPS	6	Modifier relationship	MOD
7	Apposition relationship	APP	8	Quantitative relationship	QUN
9	Constellation relationship	COO	10	Connection relationship	CON
11	Referential relationship	REF	12	Qualified relationship	DET
13	Negative relationship	NEG	14	Interrogative relationship	ITG
15	Location relationship	LOC	16	Time relationship	TMP
17	Expression relationship	EXP	18	Genitive relationship	GEN
19	Ergative relationship	ERG	20	Dative relationship	DAT
21	Comitative relationship	COG	22	Plural relationship	PLU
23	Honorific relationship	HON	24	Nominalized relationship	NML
25	ROOT relationship	ROOT	26	Tense and aspect relationship	TAM
27	Punctuation relationship	PUN	28	Description relationship	DES
29	Particle relationship	PAR	30	Target relationship	TAR
31	Auxiliary relationship	AUX	32	Manner relationship	MAN
33	Source relationship	SOU	34	Non-predicative verb relationship	PER

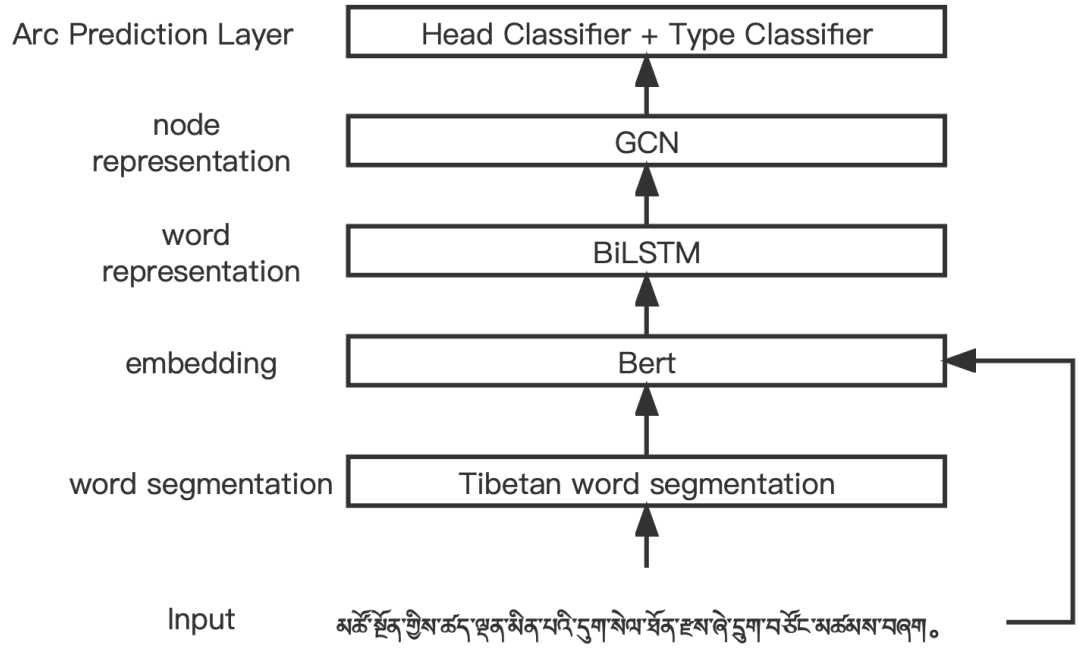

Figure 2: The framework of Tibetan dependency parsing.

Table 2: The statistics of the dataset.

Dataset	#Instances	Average Length
train	4970	6.8
validation	200	6.7
test	400	7.1

Finally, the head node with the highest score is selected as per Equation (4) to correspond to the word.

$$head_i = \max_{0<x<n} h_x \tag{4}$$

Type Classifier. The number of dependencies between head and dependent words is fixed, making this a fixed-class classification problem. To better model the dependency relationship between the head and dependent words, we use both the representations of the head and dependent words to predict the type of dependencies, as shown in Equation (5). Where j is the head word index of word i; $\mathbf{W_1} \in \mathbf{d^{f*m*f}}$ and $\mathbf{W_2} \in \mathbf{d^{2f*m}}$ a parameter matrix; m is the number of types of dependency arcs; $\vec{b} \in d^m$ is a parameter vector.

$$head_i^{label} = \vec{f_j^T}\mathbf{W_1}\mathbf{r_i} + (\mathbf{r_j} \oplus \mathbf{r_i})\mathbf{W_2} + \mathbf{b} \tag{5}$$

The type of the dependency arc is predicted using Equation (6).

$$type_i = \max_{0<x<m} head_x^{label} \tag{6}$$

4 Experiments

In this section, we conduct experiments of Tibetan dependency analysis task on the dataset from (An and Long, 2021) .

4.1 Dataset

The dataset is divided into three parts: the training set, validation set, and test set. Table (2) displays the statistics of the dataset.

4.2 Evaluation Metrics

Four evaluation metrics are employed in this paper, as follows.

4.2.1 UAS

The unlabeled attachment score (UAS) is defined as the percentage of all words that have found their correct head word, including the root node. Notably, this metric does not consider the type of dependency arcs. UAS is calculated as per Equation (7), where N_{word}^{head} is the number of words labeled with the correct head word, and N_{word} represents the total number of words in the dataset.

$$UAS = \frac{N_{word}^{head}}{N_{word}} \tag{7}$$

4.2.2 LAS

The labeled attachment score (LAS) is defined as the percentage of all words that have the correct head word and the correct type of dependency arc, including the root node. LAS is calculated as per Equation (8), where N_{word}^{arc} is the number of words with the correct head word and type of dependency arc.

$$LAS = \frac{N_{word}^{arc}}{N_{word}} \tag{8}$$

4.2.3 UEM

The unlabeled exact match score (UEM) is defined as the percentage of sentences in which all the words have the correct head words. UEM is calculated using Equation (9), where $N_{sentence}^{head}$ is the number of sentences with correct head words for all their words, and $N_{sentence}$ is the total number of sentences in the dataset.

$$UEM = \frac{N_{sentence}^{head}}{N_{sentence}} \tag{9}$$

4.2.4 LEM

The labeled exact match score (LEM) is defined as the percentage of sentences in which all the words have the correct head words and type of dependency arcs. LEM is calculated using Equation (10), where $N_{sentence}^{arc}$ is the number of sentences with correct head words and type of dependency arcs for all their words.

$$UEM = \frac{N_{sentence}^{arc}}{N_{sentence}} \tag{10}$$

4.3 Experimental Settings

The validation set is utilized to determine the best hyperparameters for the model. The hyperparameters of the model are then set as follows: Tibetan-Roberta-base[1] is employed to

[1]https://huggingface.co/sangjeedondrub/tibetan-roberta-base

generate the embedding of Tibetan syllables, and the vectors are composed into word embeddings based on BiLSTM. The dimension of the word embedding is set at $d^w = 768$, whereas the dimension of the semantic role label is set at $d^l = 100$.

The dimension of the multi-layer perceptron matrix MLP^{head} is set at $768 * 100$, and the dimension of MLP^{dep} is also set at $768 * 100$. The model's dropout is set at 0.3, and it employs the adadelta optimizer, with the learning rate set at 0.001. All parameters are randomly initialized using a uniform distribution among $[-0.2, 0.2]$.

Moreover, we compare our model with deep learning-based Tibetan dependency parsing models, including word2vec + BiLSTM + MLP (DL-BiLSTM), word2vec + RNN + MLP (DL-RNN), word2vec + GRU + MLP (DL-RNN) and word2vec + Stacked LSTM + MLP (DL-Stacked) from (An and Long, 2021). And the word embedding is trained by fastText (Thavareesan and Mahesan, 2020). In addition, we compare our model with R-GCN (Schlichtkrull et al., 2018) with similar settings.

All experiments were conducted on a GPU server with a CPU configuration of 2* AMD Skyline 7742, 512G DDR4 RAM, and 4* Nvidia A100 40G GPU cards.

4.4 The Overall Experimental Results

The Tibetan dependency parser is designed to predict the head word and type of dependency arc for each word in a sentence. Our framework is implemented using Pytorch, and the overall results are presented in Table (4).

The experimental results demonstrate that the method proposed in this paper achieved the best performance on all four metrics, highlighting the value of graph neural networks in Tibetan dependency analysis. And our proposed method achieves better performances than R-GCN, we speculated that R-GCN requires more data to train the relation representation matrix, whereas our training data is sufficient to train the relation matrix effectively.

4.5 Ablation Study

To better understand the impact of different parts of the model on the experimental results, an ablation study was conducted to analyze the value of pre-trained language models and graph neural networks in Tibetan dependency analysis. The experimental results are presented in Table 3, where "- GCN + LSTM " represents our proposed model replacing GCN with LSTM, "-Bert + GCN" represents our proposed model replacing Bert with word2vec.

From these results, two conclusions can be drawn: (1) Graph neural networks significantly impact the performance of Tibetan dependency analysis, and using word embeddings as the lexical representation method can still improve the performance. (2) Tibetan pre-trained language models also hold value in Tibetan dependency analysis, and the performance of Tibetan dependency analysis declines to some extent when using word embeddings to replace the BERT model.

5 Conclusion

To address the issue of inadequate modeling of dependency graph information in current Tibetan dependency analysis methods, this paper proposes a graph neural network-based approach for Tibetan dependency analysis. Furthermore, a Tibetan pre-trained language model is employed to improve the performance further. The experimental results demonstrate the effectiveness of the graph neural network and the Tibetan pre-trained language model for enhancing Tibetan dependency analysis. Large models such as ChatGPT have recently achieved significant results in natural language processing tasks such as dialogue and knowledge extraction. In the future, we aim to explore large models for low-resource languages and their potential applications in low-resource scenarios.

6 Acknowledgments

This work is supported by the Natural Science Foundation of China (22BTQ010), the National Natural Science Foundation of China (62076233) and the Innovation Project major research of Chinese Academy of Social Sciences (2022MZSQN001).

References

Bo An and Congjun Long. 2021. Neural dependency parser for tibetan sentences. *Transactions*

Table 3: Overall results of various models on Tibetan dependency parsing datasets.

Model	UAS	LAS	UEM	LEM
DL-RNN	0.905	0.851	0.675	0.54
DL-GRU	0.913	0.865	0.677	0.565
DL-LSTM	0.918	0.867	0.687	0.560
DL-Stacked	0.912	0.864	0.680	0.570
R-GCN	0.907	0.852	0.630	0.542
Our model	**0.924**	**0.879**	**0.696**	**0.577**

Table 4: The result of ablation study.

Model	UAS	LAS	UEM	LEM
Our model	**0.924**	**0.879**	**0.696**	**0.577**
- GCN + LSTM	0.913	0.865	0.677	0.565
-Bert + GCN	0.918	0.867	0.687	0.560

on Asian and Low-Resource Language Information Processing, 20(2):1–16.

Daniel Andor, Chris Alberti, David J Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. *arXiv: Computation and Language*.

Qingxing Cao, Xiaodan Liang, Bailin Li, and Liang Lin. 2019. Interpretable visual question answering by reasoning on dependency trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. pages 740–750.

Hao Cheng, Hao Fang, Xiaodong He, Jianfeng Gao, and Li Deng. 2016. Bi-directional attention with agreement for dependency parsing. *arXiv: Computation and Language*.

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*.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Jiecairang Duo, Quecairang Hua, Keyou Huan, and Rangdangzhi Cai. 2021. Transition based neural network dependency parsing of tibetan. In *MATEC Web of Conferences*, volume 336, page 06018. EDP Sciences.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv: Computation and Language*.

Christian Faggionato and Marieke Meelen. 2019. Developing the old Tibetan treebank. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 304–312, Varna, Bulgaria. INCOMA Ltd.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv: Computation and Language*.

Quecairang Hua, Wenbing Jiang, Haixing Zhao, and Qun Liu. 2013. Semi-automatic building tibetan treebank based on word-pair dependency classification. *Journal of Chinese Information Processing*.

Weina Zhao Jian Wu Huidan Liu, Minghua Nuo and He Yeping. 2012. Segt:a practical tibetan word segmentation system. *Journal of Chinese information processing*.

Zhanming Jie, Aldrian Obaja Muis, and Wei Lu. 2017. Efficient dependency-guided named entity recognition. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *arXiv: Computation and Language*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A Smith. 2016. What do recurrent neural network grammars learn about syntax. *arXiv: Computation and Language*.

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

Hua que-cai rang and Hai Xing Zhao. 2013. Tibetan text dependency syntactic analysis based on discriminant. *Computer Engineering*, 39(4):300–304.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pages 593–607. Springer.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. *arXiv: Computation and Language*.

Tashi-Gyal and Duo-La. 2015. Theory and method of tibetan dependency treebank construction. *Journal of Tibet University*.

Sajeetha Thavareesan and Sinnathamby Mahesan. 2020. Sentiment lexicon expansion using word2vec and fasttext for sentiment prediction in tamil texts. In *2020 Moratuwa engineering research conference (MERCon)*, pages 272–276. IEEE.

Nima-Zhaxi Toudan Cairang and Wanme Zhaxi. 2018. Study on the technique of tibetan dependence treebank building. *Plateau Science Research*, 2(03):103–109.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020. Amr-to-text generation with graph transformer. *Transactions of the Association for Computational Linguistics*, 8:19–33.

XIA Wuji and HUAQUE Cairang. 2019. Dependency tree based tibetan semantic dependency analysis. *Journal of Tsinghua University (Science and Technology)*, 59(9):750–756.

Zou Xiaomei, Yang Jing, Zhang Jianpei, and Han Hongyu. 2018. Microblog sentiment analysis with weak dependency connections. *Knowledge-Based Systems*, 142:170–180.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81.