ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **26**, Number 1, 2023, 3–20

A Frequent Construction Mining Scheme Based on Syntax Tree

Bob CHEN , Weiming PENG , and Jihua SONG*

School of Artificial Intelligence, Beijing Normal University, Xinjiekouwai St 19, Haidian District, 100875 Beijing, China

E-mails: bochen@mail.bnu.edu.cn, pengweiming@bnu.edu.cn, songjh@bnu.edu.cn*

* Corresponding author

Abstract. *Natural language processing* (NLP) is one of the main research directions in artificial intelligence. One of the goals of NLP is to identify various semantic information in the text. Currently, the mainstream semantic recognition tasks focus more on using the semantic information of each word in the text to perform semantic analysis of the entire sentence. The research on semantics in cognitive linguistics indicates that semantics is determined by both the words contained in the sentence and the arrangement of the words. Linguists refer to permutations and combinations containing certain semantic information as constructions. Since the construction plays an essential role in semantic information, identifying various constructions in text is a crucial work of semantic recognition tasks. Based on this background, the main works performed in this paper are as follows: 1) The definition and program representation of constructions and the corresponding constraints in NLP tasks are proposed. 2) A frequent construction mining algorithm is proposed to extract frequent structures that meet the construction requirements in the grammar structure tree. Based on the above works, the corresponding construction database can be extracted for the specified natural language corpus, which is helpful for more effective text semantic analysis.

Key-words: Construction; data mining; semantic recognition; sequential pattern mining.

1. Introduction

Natural language processing (NLP), as one of the main research directions in artificial intelligence, aims to identify various semantic information in the text. Currently, many algorithmic models attempt to parse the semantic information contained in sentences. However, most of the existing algorithms are based on the assumption that the semantic information is determined by the words that make up a sentence. For example, sentence embedding can be obtained by adding

word embedding. This often loses many details of the statement itself. For example, the sentence "What's the fly doing in my soap?" is interrogative if understood only from the surface. However, the main purpose of the questioner is not to know the specific reason, but to question or satirize this rare phenomenon through this rhetorical question. Conventional analysis methods cannot easily identify this latent semantic information.

Cognitive linguists put forward the concept of constructions to solve the semantic recognition of sentences [1, 2]. The related research results indicate that the construction determines the core semantics. The so-called construction is a form-meaning pair: the arrangement of the components in the sentence [1, 2]. The semantics themselves are mainly performed through the arrangement of words or phrases. Taking the statement "What's the fly doing in my soap?" as an example, it contains one of the main constructions, which is "What's [] doing []". This construction itself carries pragmatic information like questioning and sarcasm. The square brackets represent an empty slot, and no matter where the empty slot is filled with an acceptable word, the composed sentence contains a kind of questioning and ironic meaning. For example, "What are you doing in my room?" or "What's the cat doing on the table?". These sentences all contain questioned semantics, and this part of the semantic information is irrelevant to the specific words filled in the empty slots. The filled-in words should associate these semantics with the actual scene.

The first step of the above semantic analysis idea is to identify common constructions from many corpora. Common constructions are extracted according to the analysis results of Stanford Parser. The output of Stanford Parser is a tree structure, with frequent subtree mining and frequent subgraph mining algorithms. However, there are many differences between subtrees or subgraphs and constructions. For example, the extracted construction in the above example should be "What's NP/NN doing PP/NP", which is neither a subtree nor a subgraph in the syntax tree. Among them, "What's" is not adjacent to "doing" and cannot form a subtree or subgraph with the "NP/NN" part. Therefore, a new algorithm should be established to identify and express its corresponding construction information. This problem is defined as frequent construction mining.

Based on the above background, the construction recognition problem is defined as the problem of frequent construction mining based on the corpus-generated syntax tree database. Besides, it is necessary to redefine and constrain its frequent structures. This paper mainly performs the following works: 1) The definition and program expression of constructions in NLP tasks and the corresponding restrictions are proposed; 2) The frequent construction mining algorithm is proposed to extract frequent structures that meet construction requirements in the tree structure. Accordingly, the corresponding construction database can be extracted for the specified natural language corpus. Ultimately, through this information, we can mine richer semantic information that cannot be identified by traditional methods.

2. Basic Concepts and Problem Statement

This section defines the concepts involved in the construction mining task and illustrates the goal of the construction mining task with examples.

2.1. Basic concepts

Definition 1: Construction

There is still some controversy over the definition and scope of constructions in linguistics. This paper considers the construction as a form-meaning pair at the sentence level composed of two or more symbolic units. The construction is divided into the following three forms. 1) Fixed idiom construction, such as all of sudden, going great guns, facing the music, and other idioms, in which each word is fixed. 2) Semi-fixed idiom constructions, such as send _ to the cleaners and jog _ memory. Although the whole sentence may not follow the regular syntactic rules, it can also carry a part of semantics. 3) Sentence construction like What's _ doing? and The _er the _er, where the empty slots can be words or phrases. Its fixed part can be a word or a generalized unit type such as "NN" and "NP" [3]. This paper maps constructions to structural sequences. In other words, structures are arranged in syntactic units in natural sentences.

Definition 2: Constant

In the construction, the invariable syntactic unit is the constant. For example, in the construction What's _ doing _ ?, "What's" and "doing" are constants. The construction consists of a constant and empty slot.

Definition 3: Empty slot

An empty slot is a portion to be filled in the construction. Empty slots allow different words or phrases to be filled in and provide ductility and productivity to the construction. The number of empty slots in each construction is indeterminate, such as the idiom "all of a sudden" does not contain an empty slot. The construction "what's x doing y?" contains two empty slots. Square brackets are employed in this paper to denote empty slots, and the construction "what's x doing y?" can be expressed as what's [] doing [].

Definition 4: Phrase structure tree

The phrase structure tree is utilized to express the syntactic structure, and each node in the tree represents a syntactic unit in the sentence. The tree's leaf nodes are the specific words appearing in the sentence. The intermediate nodes are various phrase components related to the corresponding word. For example, the phrase structure tree of the sentence: "The girl took apples from a bag". is shown in Fig. 1, where S represents the entire sentence, while each leaf node is a word in the sentence, other nodes such as "DT" represent Determiner, and "NP" stands for Noun Phrase.

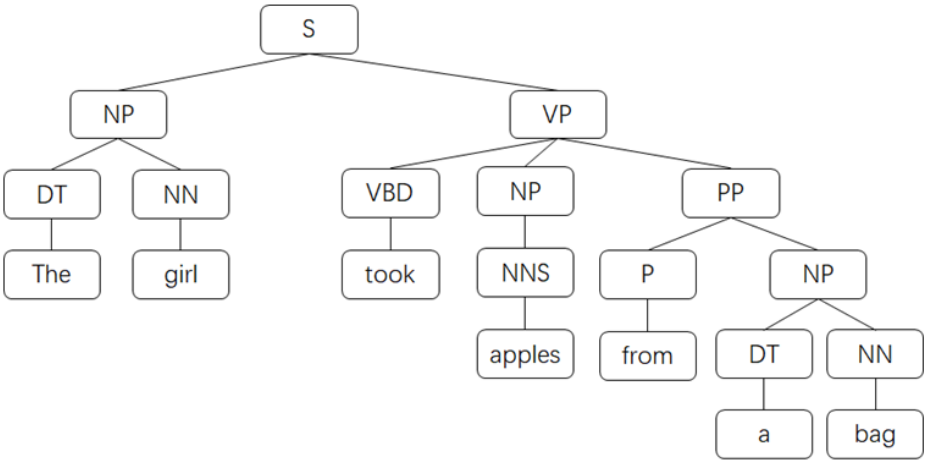

Fig. 1. Phrase structure tree.

Definition 5: Tree sequence

In order to extract frequent structures from phrase structure trees, tree sequence is defined as a new concept of sequence. The tree sequence is extracted from the entire tree structure. It contains specific words or other syntactic units. For example, from Fig. 1, the sequences like "DT girl VP". and "The girl took NP PP." can be extracted. For specific extraction methods, please refer to Section 4.2.

Definition 6: Support

The support of a sequence α is the proportion of a sequence in the database S, denoted as Support (α).

Definition 7: Frequent construction Frequent constructions are sets of constructions that frequently appear in a corpus. Given the minimum support threshold ξ, if the support of the construction α in the corpus is not lower than ξ, the sequence α is called a frequent construction.

2.2. Problem statement

This paper aims to solve the following problems:

The input is a natural language corpus, minimum support. After corpus preprocessing, algorithm processing, and other steps, the final output is the set of frequent constructions appearing in this corpus.

For example, a corpus is given as follows:

What's the fly doing in my soap?

What's the cat doing in my table?

What's the boy doing in my room?

The girl took apples from a bag.

The minimum support is 2. Then it produces frequent constructions such as What's [NP] doing [PP], where square brackets represent an empty slot, and its content is the structural unit corresponding to the empty slot. The example "The girl took apples from a bag". contains the construction "[NP] took [NP] [PP]". However, since the mentioned construction does not satisfy the minimum support, "[NP] took [NP] [PP]" is an infrequent construction.

3. Related Work

This paper deals with mining frequent constructions from phrase structure tree, a tree structure, and a special graph structure. Thus, the work of this paper is related to graph mode and tree mode. In order to solve the problem of frequent subgraph mining, the earlier *Apriori-based Graph Mining* (AGM) algorithm [4] employed the Apriori algorithm for reference in matrix calculation and found frequent subgraphs in the graph data set layer by layer through the fixed-point growth method. Currently, the most widely used gspan algorithm [5] improves the operation efficiency of the algorithm by expanding the traversed fixed-point set to construct a DFS tree. In the later stage, various gspan-based extended algorithms have been proposed to improve the operation efficiency of the algorithm and deal with different subgraph mining problems. The fixed size *subgraph sampler* (FS) [6] algorithm searches for approximate top-k frequent subgraph sets and performs frequent subgraph sampling in the fixed-size subgraph space to avoid the subgraph isomorphism calculation. Accordingly, it can improve running speed significantly. However, there is the possibility of returning to infrequent patterns. Recent algorithms like k-Fismw [7] and TKG [8] have improved the algorithm's running speed from different aspects.

Another relatively basic field in pattern recognition tasks is frequent sequential mining, including Breadth-first search algorithms represented by AprioriAll [9] and GSP [10], Depth-first search algorithm represented by Spade [11], Spam [12], CM- Spam [13], and Fast [14], and Pattern-growth algorithm represented by FreeSpan [15] and PrefixSpan [16]. Based on these algorithms, many optimization and pruning strategies have been proposed to improve the efficiency of algorithm execution. Various works were performed to mine more valuable sequences,

such as weighted sequential pattern mining [17-19] and high-utility sequential pattern mining [20]. These concepts were introduced to mine more valuable information from real scenarios in the original sequence database. For example, according to weighted sequential pattern mining, the set of elements in the sequence database should contain different weights and should not be generalized. Based on the high-utility sequential pattern, the effect of quantity on the set of elements in the sequence should be considered.

The problem raised in this paper implies parsing the corpus into a phrase structure tree containing syntactic unit information. Here, Stanford Parser is employed to handle this process. Stanford Parser provides a syntactic parsing framework and the *Probabilistic Context-Free Grammar* (PCFG) algorithm [21]. PCFG combines left-most derivations and different rule probabilities in *context-free grammars* (CFG) to calculate all possible tree structure probabilities and takes the tree corresponding to the maximum value as the syntactic analysis result of the sentence. Stanford Parser can well solve the grammar parsing problem.

The phrase structure analysis model mainly includes the syntactic analysis model based on the transition system, which restores a syntactic tree by predicting the sequence of actions to generate the syntactic tree. References [22, 23] employed the syntactic analysis model based on dynamic programming to recursively predict the sub-phrase with the highest score for each phrase and finally backtrack to restore the optimal syntactic tree. Reference [24] established a sequence-to-sequence syntactic analysis model to map the syntactic tree into a unique corresponding sequence representation and predict sequences by sequence labeling or sequence generation [25].

Some researchers try to add the structural features of sentences in the process of semantic analysis. Tree-LSTM introduce a new LSTM architecture to mining structural features [26]. Syntax-BERT integrate syntax trees into pre-trained Transformers [27]. These models use neural networks to automatically mine structural features. Compared to these models, the method presented in this paper explicitly extracts structural features.

At present, for the proposed structural pattern mining, there are no related algorithms and studies that can be directly applied from the research fields of graph pattern, tree pattern, sequential pattern, and syntactic tree parsing. Therefore, improved algorithms should be proposed based on relevant models in various fields to solve the construction mining problem.

4. The Method

In order to solve the problem of frequent construction extraction, the natural corpus should be converted into a form that contains more syntactic component information, such as a phrase structure tree, to satisfy the conditions for forming constructions. Moreover, a new pattern mining method should be proposed to mine the frequently occurring constructions in the corpus. Then, corresponding pruning and screening strategies are formulated to filter the selected pattern patterns further to simplify the construction set. Finally, the storage structure required by the construction is defined, and the corresponding frequent construction set is generated through the preset threshold. The specific steps are described in the following subsections: 1. Preprocessing input data; 2. Tree sequence extraction; 3. Sequential mining algorithm; 4. Calculation of the probability of construction empty slot items; 5. Filtering and screening of construction.

4.1. Input and output data format definition/corpus preprocessing

At present, there are three main ways to analyze natural sentences. 1) Phrase structure parsing is to identify the phrase structure in the sentence and the hierarchical syntactic relationship between the phrases. 2) Dependency syntactic parsing is to identify the interdependence between words in a sentence. 3) Syntactic analysis of deep grammars, that is, using deep grammars, such as *Lexicalized Tree Adjoining Grammar* (LTAG) [28], *Lexical Functional Grammar* (LFG) [29], and *Combinatory Categorial Grammar*, CCG) [30] to perform deep syntactic and semantic analysis of sentences. According to the definition and description of constructions in cognitive linguistics, constructions mainly include specific words, word types, and phrase types. The constructions are hierarchical, which means that the generalization levels of the meanings represented by different constructions are different. Phrase structure analysis methods meet this requirement. For example, Fig. 1 shows the obtained syntax tree of the natural text "The girl took apples from a bag". after analyzing the phrase structure. The phrase structure tree contains the specific words that make up the sentence and the syntactic component information corresponding to each word, like NN and NP. Moreover, the information is also related in a tree form. It is helpful to conduct further structural mining and screening. Therefore, the phrase structure analysis method is chosen to preprocess the natural corpus.

Stanford Parser is currently the most mature and stable phrase structure analysis tool [31]. Since the main goal of this paper is to extract common structural patterns from existing structures rather than improving the accuracy of phrase structure analysis, Stanford Parser is directly adopted to process the phrase structure of the natural corpus.

4.2. Tree sequence extraction

The targets that should be extracted can cover multiple levels in the syntactic structure. For example, for the phrase structure tree in Fig. 2, frequent constructions should be mined, such as "DT girl VP." and "The girl took NP PP.". Unlike frequent subgraphs or frequent subtrees, components of frequent structures may not be adjacent in a phrase structure tree. For example, the position of "DT girl VP." in the overall phrase structure tree is shown in Fig. 2. In this case, the tree structure cannot be directly mined.

In this case, we prepare to split the phrase structure tree in the first step for obtaining several subsequences, called the tree sequence set. Two splitting methods are proposed here, and the first one is as follows:

1) Firstly, process the root node of the phrase structure tree; replace the root node with child nodes, *i.e.*, S->NP, VP; the replacement result gives a tree sequence record <{NP}, {VP}>.

2) Deeply traverse its child nodes and replace each node with child nodes: NP->{DT}{NN}; output a tree sequence record <{DT}{NN}{VP}>.

3) Continue to traverse the nodes until the leaf node: DT->{The}; output tree sequence record: <{The}{NN}{VP}>.

Since this splitting method generates a tree sequence for each child node replacement, every possible sequence combination in the structure tree is recorded in detail. This is called an expanded split. Expanded splits can be handled in the subsequent structural pattern exploration process. However, this method will split various tree sequence sets from a phrase structure tree, increasing the calculation amount of construction mining and significantly affecting the operational efficiency.

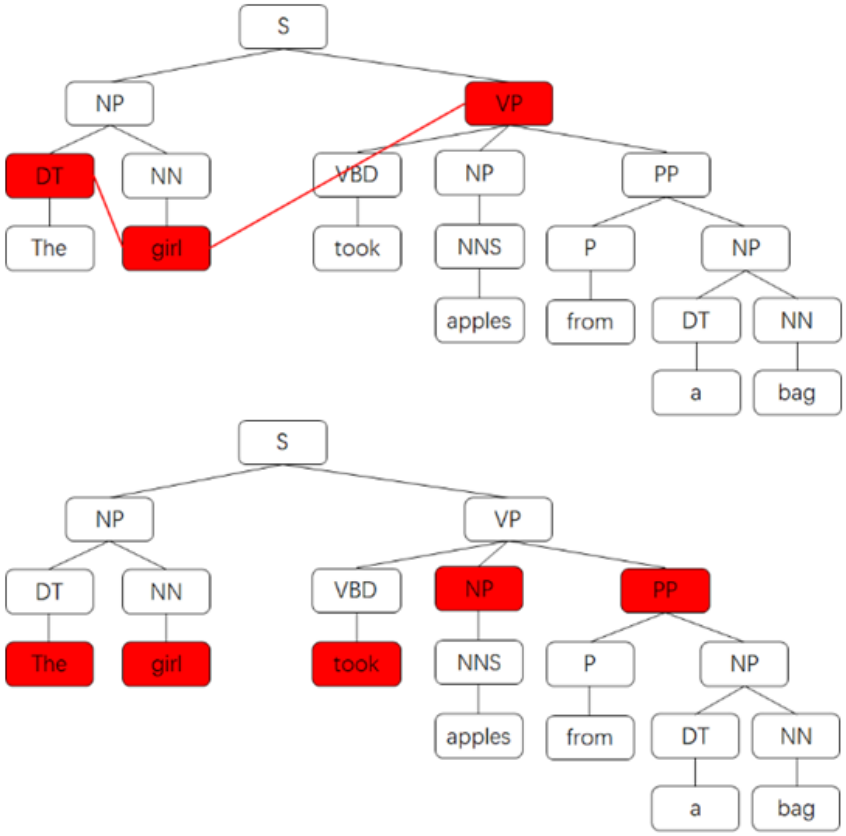

Fig. 2. Constructions in the phrase structure tree.

In order to solve the problem of tree sequence scale explosion, this paper proposes another splitting method based on the expansion splitting algorithm, namely merge splitting. The process is as follows:

1) Traverse the phrase structure tree; if a node contains only one child node, merge the two nodes to obtain the combined phrase structure tree, as shown in Fig. 3.

2) Process the root node of the phrase structure tree; replace the root node with child nodes: S->NP, VP; the replacement result outputs a tree sequence record <{NP},{VP}>.

3) Deeply traverse the combined phrase structure tree and replace each node with child nodes, *e.g.*, NP->{DT,"The"}{NN, "girl"}; output a tree sequence record <{DT,"The"}{NN,"girl"}{VP}>.

Combined splitting can significantly reduce the number of tree sequences splitting from the phrase structure tree and reduce the computational complexity of structural pattern mining. Specific words are enclosed in double-quotes to distinguish specific words from syntactic component attribute labels. However, in the subsequent execution of the mining algorithm, the sequence containing multiple sets will still be processed, increasing the running complexity of the mining algorithm.

A phrase structure tree can be converted to several tree sequences through the above two methods. In this way, the pattern mining problem for tree structure is transformed into a sequen-

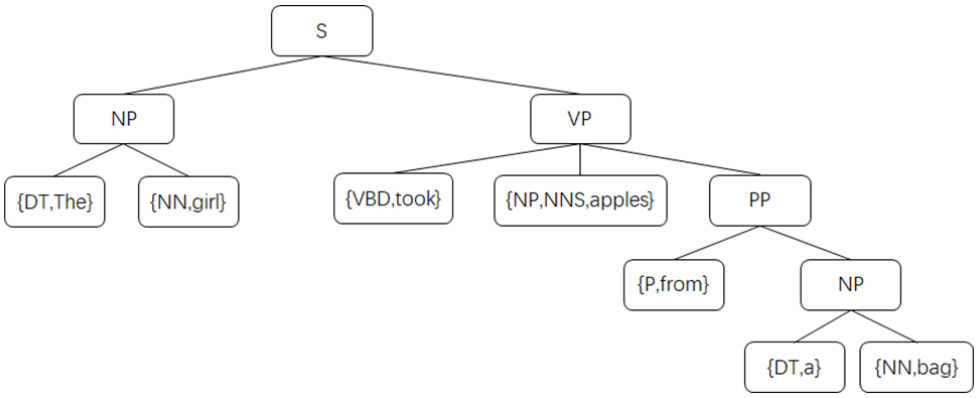

Fig. 3. The merged phrase structure tree.

tial pattern mining problem.

4.3. Sequential mining algorithm

Unlike the traditional sequential pattern mining problem, since the construction is composed of constants and empty slots, the gap information should be retained in the frequent pattern, which corresponds to the empty slot part in the construction. In order to solve the above problems, the existing algorithm is generalized to increase the processing of empty slots. Prefixspan is a typical Pattern-growth algorithm, which has the following three characteristics. a) The projection database is applied to perform recursive query layer by layer. b) The traversal of the algorithm is essentially a Depth-first search, in which the projection database is generated based on the sequence database. Thus, the frequent sequential patterns generated by this algorithm all exist in the sequence database. c) The algorithm logic is clear, simple, and easy to expand. Based on the above three characteristics, this paper performs the transformation based on the Prefixspan algorithm so that the generated sequential pattern contains information about the number of interval item sets.

1) Input parameter description

The original Prefixspan algorithm contains the following two parameters: original sequence database and minimum support. Extended algorithms also add gap length constraints and frequent sequence length constraints. This section does not consider extended constraint parameters. The minimum support value is employed in the algorithm to filter the frequent and interval item sets uniformly.

2) Projection database distance record

One of the innovations in the Prefixspan algorithm is the introduction of the projection database concept. A projection database is constructed for different prefixes to realize projecting a search sequence and effectively reduce the search scale of the sequence database. Moreover, an extended algorithm introduces pseudo-projection to reduce the memory footprint while searching. In order to record and process the interval item set, it is necessary to record the number of deleted interval items while constructing the projection database and accumulate the deleted interval items. Thus, the intervalItemCount variable of type Integer is added in the pseudo-projection structure to record the number of interval items.

3) Interval item set processing In the first step of the Prefixspan algorithm, the prefix elements of length 1 whose occurrences exceed the minimum support are first filtered, and the original sequences in the sequence database are then processed according to the selected prefix elements. a) Sequences that do not contain frequent prefixes are removed; b) Infrequent elements in the sequence are deleted. Our algorithm accumulates the corresponding intervalItemCount value while scanning the sequence and deleting infrequent items. In this way, different sequential patterns will be produced according to different numbers of interval item sets. For example, if the original sequence is <{a}{b}{cd}{e}>, where b and cd are infrequent elements, the processed sequence is <{a}[2]{e}>. The "2" in square brackets represents two gaps between {a}{e}. Similarly, storing the number of deleted infrequent item sets is also necessary while constructing a projection database.

4) Traversing the number of interval item sets

After obtaining frequent prefixes, a projection database should be constructed for each frequent prefix. Recursive processing is performed on the projection database of each prefix. In each recursive process, a new frequent prefix machine projection database is generated, and frequent prefixes generated by recursion are combined to generate the final frequent sequence.

The processing steps should be increased for the number of interval item sets before each recursion. We first calculate the number of interval item sets processed this time before each recursion, then loop through the values of each interval item set. In addition to passing the projection database and prefix information into the next-layer recursive function, the number of interval item sets should be passed corresponding to this projection.

For example, for the sequence database:

<{A}{C}{A}>

<{BG}{A}{DG}{AC}>

<{F}{A}{C}{F}>

<{A}{C}{A}{C}>

where {A} is the frequent item set, and the frequent sequence <{A}> is the output. By constructing the projection database of {A}, the projection database is obtained as:

[0]{C}{A}

[1]{AC}

[0]{C}

[0]{C}{A}{C}

where the frequent item set is [0]{C}, and the frequent sequence <{A},[0],{C}> is the output. After constructing the projection database of [0]{C} in the next step, the projection database is obtained as:

[0]{A}

Empty

Empty

[0]{A}{C}

At this time, the frequent item set [0]{A} is obtained, and the frequent sequence <{A},[0],{C},[0],{A}> is the output. There is no new frequent item set in the projection database, and the algorithm returns. The flow chart of the algorithm is shown in Fig. 4.

5) Output processing

In the recursive traversal process, after generating a frequent prefix, this prefix is combined with the prefix generated before forming a new frequent subsequence. The number of interval

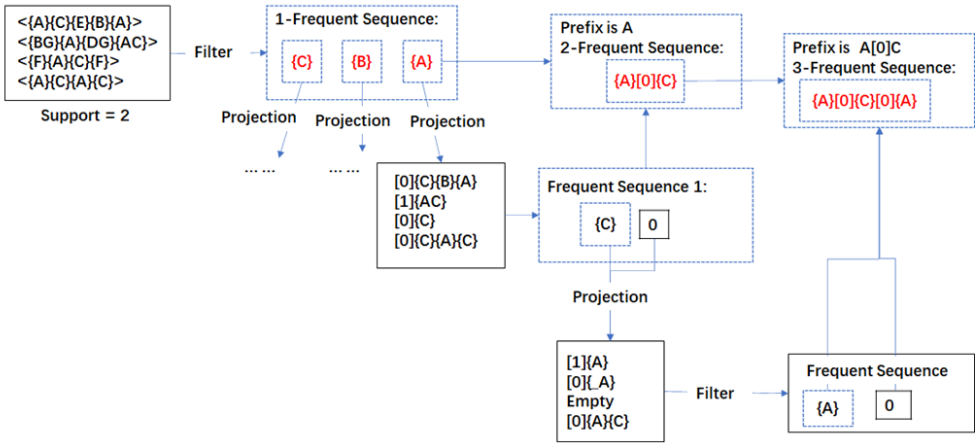

Fig. 4. Flow chart of the construction mining algorithm.

item sets passed recursively is output to form a sequential pattern containing the number of interval items during the generation process.

6) Algorithm description

The overall ConstructionMinning algorithm flow is as Figure 5:

Input: A sequence database S, and the minimum support threshold *min_sup*
Output: The complete set of sequential patterns
Method: Call ConstructionMinning(‹›, 0, S, ‹›)
Subroutine ConstructionMinning(α, l, $S|_\alpha$, i)
Parameters: α: a sequential pattern; l the length of α; $S|_\alpha$: α-projected database, $\alpha \neq$ ‹›;otherwise, the sequence database S. i:the interval item count before α, $\alpha \neq$ ‹›;
Method:
1. Scan $S|_\alpha$ once, find the set of frequent items b such that
(a) b can be assembled to the last element of α to form a sequential pattern; or
(b) ‹b› can be appended to α to form a sequential pattern.
2. For each frequent item b and interval items count i, append them to α to form a sequential pattern α', and output α';
3. For each frequent item b, scan $S|_\alpha$ for finding all interval items count which the count larger than *min_sup*. Build frequent interval items count f;
4. For each f and α', construct α'-projected database $S|\alpha'$, and call **ConstructionMinning**(α', l+1, $S|\alpha'$, f)

Fig. 5. Pseudocodes of construction mining.

4.4. Calculation of the probability of construction empty slot items

After obtaining the sequential pattern containing the number of interval items, the item type preference should be calculated corresponding to each empty slot in the sequential pattern. For example, the following information should be obtained. For the construction "What's [] doing []", the probability of occurrence of NP in empty slot 1 is 76%, that of NN is 21%, and that of other types is 3%. In this way, the filling constraints of the construction on its hollow slot can be derived.

In order to achieve the above goals, the tree sequence database is traversed again after performing the sequential pattern mining. For the sequential patterns that conform to the construction, the number of filled items is counted separately and divided by the total number of constructions to obtain the item preference probability of each empty slot in the construction.

4.5. Filtering and screening of construction

1) Screening of construction abstraction

The above steps result in a set of structural patterns. One of the problems is that for the construction "What's [] doing []", there will be another more abstract construction, such as "WHNP's [] VBG []".WHNP is the component type of the word "What", if this word is a frequent item, WHNP must also be a frequent item. However, based on the theoretical analysis of cognitive linguistics, "What's [] doing []" is a semi-fixed idiom construction, while "WHNP's [] VBG []" is not the construction. Therefore, we should remove this repetitive pattern caused by ingredient labels. Another threshold, MaxRepeat, is set for the component labels. The main function is to filter the sequential pattern if the ratio of a specific item to the corresponding component label item in the output construction set exceeds MaxRepeat.

2) Construction pruning and filtering

Some extensions can filter frequent sequences by limiting the gap length when the Prefixspan algorithm produces frequent sequences. On the one hand, this can produce more meaningful sequential patterns because if the gap is too large, the two sets of elements may be unrelated even if two item sets are adjacent. On the other hand, limiting the gap length also plays a certain role in pruning. The MaxGap value is set through human experience and applied to the algorithm at the beginning of the execution. According to the research background of this paper, if the two syntactic components are too far apart, it can be judged that the combination is not the construction, even if the combination occurs frequently. Thus, when dealing with the third step of the above algorithm, in addition to comparing the number of items in each interval with the minimum support, a condition should also be added, that is, whether the distance is greater than the preset MaxGap value. If this condition is met, the value is set to return directly. In this way, the purpose of filtering frequent sequences with large distances is realized while generating frequent sequences.

5. Experimental Evaluation

Some tests were carried out on the proposed construction extraction scheme. The test objectives are divided into two aspects: performance test and algorithm operation efficiency test. The performance test mainly evaluates the scheme's ability to extract constructions. It will test the

number of frequent output sequences and the most frequent construction examples with different input parameters, and analyze the examples combined with linguistic research.

The operation efficiency test evaluates the running speed and memory consumption of the algorithm under different input conditions. The experiments were performed on a PC with an Intel Core i7 processor and 16 GB RAM. Since our main goal is not to improve the operation efficiency of existing algorithms, it is unnecessary to make too many comparisons with existing algorithms in terms of operational efficiency indicators. Since the construction extraction algorithm is mainly based on the extension of the PrefixSpan algorithm, it is only compared with PrefixSpan in terms of efficiency, mainly to examine its impact on the operation efficiency of the original algorithm after gap information processing.

The Leipzig Corpora Collection was utilized as the original corpus. The corpus background is the 2020 English news corpus. The employed text material was taken from news websites (typically daily via RSS feeds), with 1,000,000 natural sentences. The corpus can be downloaded from https://corpora.uni-leipzig.de/. The original corpus was filtered to obtain special characters and super-long sentences, while finally, 2.900 million sentences were filtered out for experimental analysis.

5.1. Experiment 1: performance test of the construction extraction scheme

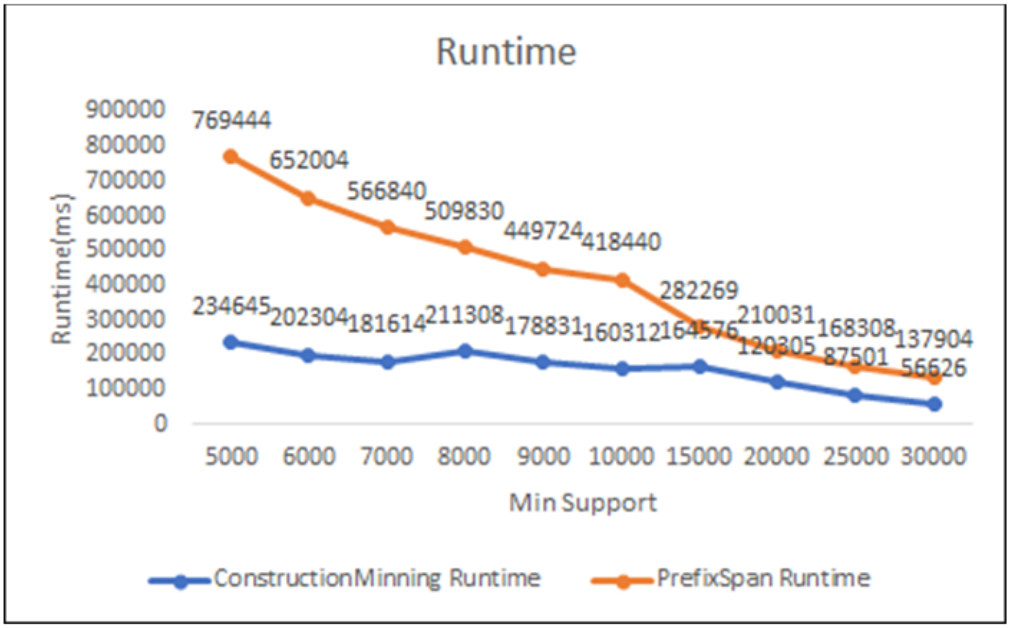

Fig. 6. Number of constructions.

Based on the proposed scheme, construction mining experiments were accomplished on the natural corpus. From the analysis of the number of experimental results, the original corpus contains 290,000 natural sentences, and 6,235,000 sentences were obtained after combined splitting

and expansion. On average, each natural text is extended by 21.5 sentences. The constructions were mined based on the expanded corpus, and the number of constructions is obtained according to the difference of minimum support, as shown in Fig. 6. Construction identification was performed for each original sentence through the construction identification procedure. According to the calculations, each natural sentence contains an average of 28.3 constructions.

Table 1 shows the examples of the most frequently occurring constructions. It can be seen that the 10 most frequently occurring constructions are all abstract constructions, that is, the constructions without specific words. According to the treatment of constructions in linguistics, these constructions play a fundamental role in the organization of language. For example, NP VP is the verb-object construction. According to cognitive linguistics analysis, the frequent use of this construction results from the human perception of the outside world in the natural environment. VP plays an organizational role in the whole construction, while NP is a participating component of VP. The remaining constructions are also the main constructions that construct the natural language. All these structures have been studied in cognitive linguistics, which is consistent with the experimental results.

Table 1. Examples of high-frequency constructions

Construction	Support
NP IN	571151
DT NN	442807
IN NP	441751
IN NN	395442
NN NP	382049
NP VP	365659
DT NP	357953
NN IN	357522
DT IN	332879
NP PP	322277

Besides, some sentences have been extracted for analysis. Take the following sentence as an example.

1) For example, one man can be a father, a son, and a husband.

The sentence includes constructions like IN NP and NP VP. It is worth noting that this sentence also contains the constructions “a” NN, “a” NN, “a” NN, which we call the a+NN parallel construction. Although there is a lack of contextual information, it is evident that the real intention of the speaker is not to explain the fact that a man can be a father, a son, or a husband. The speaker wants to say that a man can have many roles. The implied meaning of “many” is performed by the a+NN parallel construction. After searching the corpus, the following different sentences containing the a+NN parallel construction are found:

2) But money cannot replace a life, a wife, a mother, nor a friend.

3) A good man, a great writer, a great husband, and a father.

4) Phillip Weah experienced a flood, a fire, and a funeral, all in 24 hours.

The analysis of the above sentences indicates that these sentences are all statements of a fact, which contains a kind of “many” meaning. This meaning cannot be effectively analyzed if only in a literal sense. Based on the proposed scheme, the similarity of the above sentences can be identified and explained from the perspective of the construction.

5.2. Experiment 2: comparison of operation efficiency test

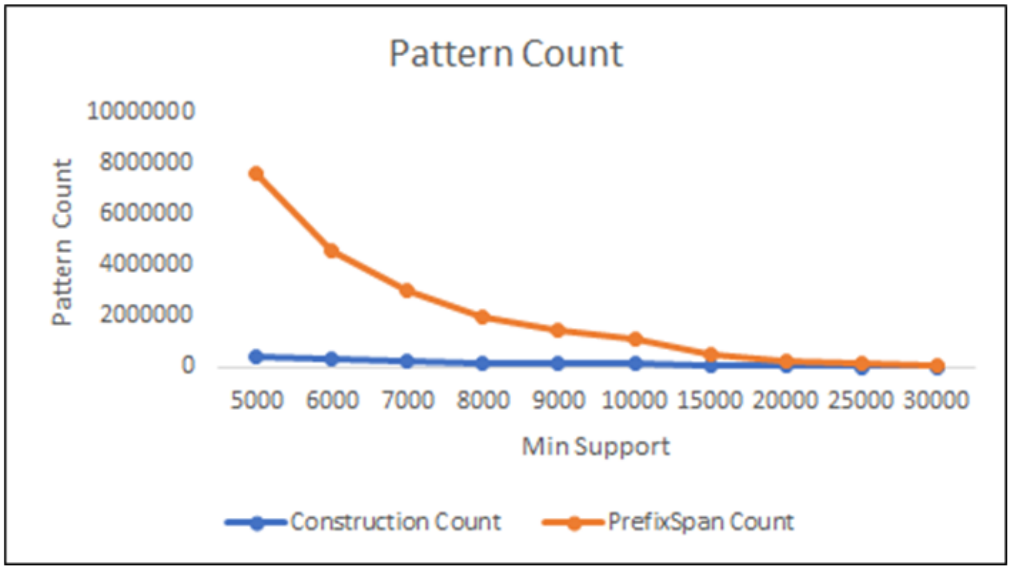

Fig. 7. Algorithm running time.

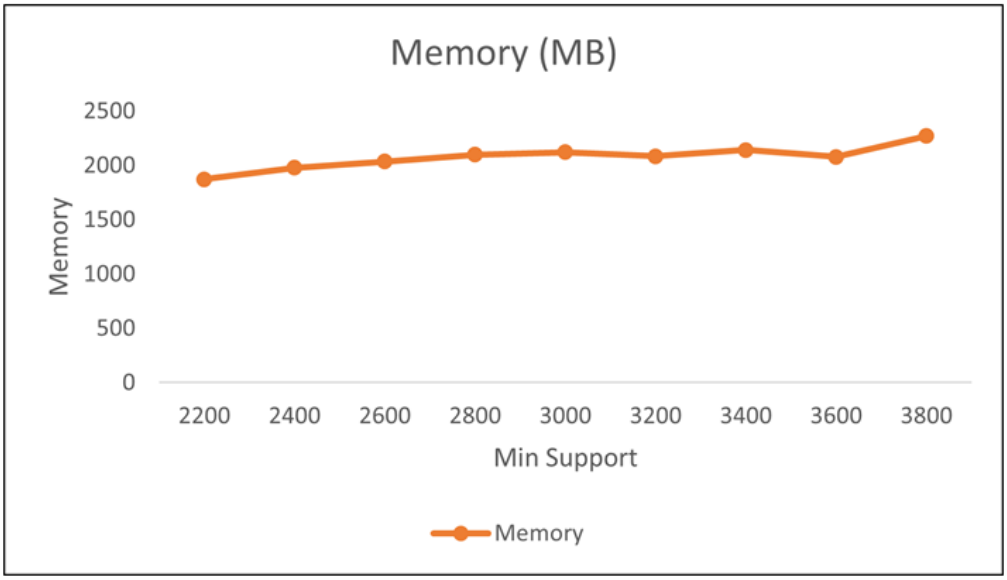

Fig. 8. Number of constructions.

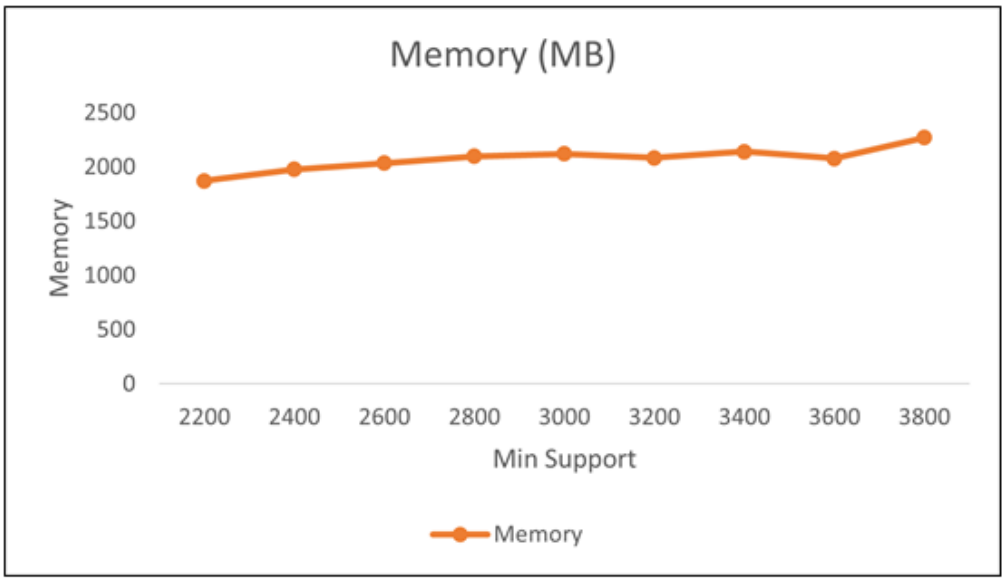

Fig. 9. Number of constructions.

This experiment investigated the effect of increasing the number of interval items on the operation efficiency of the original Prefixspan algorithm. We have tested the effect on the algorithm's running time and the number of frequent sequences produced when passing different support degrees. The experimental results are shown in Fig. 7. From the perspective of time consumption, the proposed sequential pattern mining algorithm will impact the running time, because it will traverse different item values each time. With the increase of the minimum support, the impact force becomes weaker. Fig. 8 shows the change in the number of generated sequential patterns. According to Fig. 9, since the projection database is used in the algorithm, the memory consumption is not affected by the minimum support, and the memory usage is stable. By adding the processing of different distance values, the number of frequent sequences is reduced compared with the prefixspan algorithm. With the increase of the minimum support setting value, the difference in the number of frequent sequences decreases.

6. Conclusions

This paper attempts to mine constructions in natural corpora based on construction grammar theory in cognitive linguistics. By extracting the tree sequence of the phrase structure tree and extending the frequent sequential mining algorithm, the frequent structure existing in natural language is mined. Then, the precise mining of frequent constructions is achieved by setting a series of filters and constraints. This paper introduces a new perspective of text analysis, which provides a basis for further semantic analysis based on construction theory. Accordingly, the existing word vector can be combined with the sentence vector research to conduct a more accurate semantic analysis of text sentences.

Follow-up research directions may be as follows. 1) Improving algorithm efficiency based

on other frequent sequential mining algorithms. 2) Proposing methods to combine word vector information and extracted construction information to construct a unified sentence vector expression.

Acknowledgement. This work was supported by the National Natural Science Foundation of China (Grants No. 61877004 and No. 62007004), the Major Program of National Social Science Foundation of China (Grant No. 18ZDA295).

References

[1] A. E. GOLDBERG, *Constructions: A Construction Grammar Approach to Argument Structure*, University of Chicago Press, Chicago, IL, 1995.

[2] R. W. LANGACKER, *Foundations of Cognitive Grammar: Theoretical Prerequisites*, Stanford University Press, Stanford, CA, 1987.

[3] C. J. FILLMORE, T*The mechanisms of "construction grammar"*, Annual Meeting of the Berkeley Linguistics Society, Berkeley Linguistics Society, Berkeley **14**, CA, pp. 35–55, 1988.

[4] A. INOKUCHI, T. WASHIO and H. MOTODA, *An apriori-based algorithm for mining frequent substructures from graph data*, in Principles of Data Mining and Knowledge Discovery, D. A. Zighed, J. Komorowski and J. ytkow, Eds., Springer-Verlag, Berlin, Heidelberg, pp. 13–23, 2000.

[5] X.-F. YAN and J.-W. HANY, *gSpan: graph-based substructure pattern mining*, Proceedings of 2002 IEEE International Conference on Data Mining, Maebashi City, Japan, pp. 721–724, 2002.

[6] T. K. SAHA and M. AL HASAN, *FS3: A sampling based method for topk frequent subgraph mining*, Statistical Analysis and Data Mining: The ASA Data Science Journal **8**(4), pp. 245–261, 2015.

[7] V. T. T. DUONG, K.-U. KHAN and Y.-K. LEE, *Top-k frequent induced subgraph mining on a sliding window using sampling*, Proceedings of 11^{th} International Conference on Ubiquitous Information Management and Communication, Beppu, Japan, pp. 1–7, 2017.

[8] P. FOURNIER-VIGER, C. CHENG, J. C.-W. LIN, U. YUN and R. U. KIRAN, *TKG: Efficient mining of Top-K frequent subgraphs*, in Big Data Analytics, S. Madria, P. Fournier-Viger, S. Chaudhary and P. K. Reddy, Eds., Springer International Publishing, Cham, pp. 209–226, 2019.

[9] R. AGRAWAL and R. SRIKANT, *Mining sequential patterns*, Proceedings of 11^{th} International Conference on Data Engineering, Taipei, Taiwan, pp. 3–14, 1995.

[10] R. SRIKANT and R. AGRAWAL, *Mining sequential patterns: Generalizations and performance improvements*, in Advances in Database Technology – EDBT '96, P. Apers, M. Bouzeghoub and G. Gardarin, Eds., Springer-Verlag, Berlin, Heidelberg, pp. 1–17, 1996.

[11] M. J. ZAKI, *SPADE: An efficient algorithm for mining frequent sequences*, Machine Learning **42**(1), pp. 31–60, 2001.

[12] J. AYRES, J. FLANNICK, J. GEHRKE and T. YIU, *Sequential pattern mining using a bitmap representation*, Proceedings of 8^{th} ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, AB, Canada, pp. 429–435, 2002.

[13] P. FOURNIER-VIGER, A. GOMARIZ, M. CAMPOS and R. THOMAS, *Fast vertical mining of sequential patterns using co-occurrence information*, in Advances in Knowledge Discovery and Data Mining, V. S. Tseng, T. B. Ho, Z.-H. Zhou, A. L. P. Chen and H.-Y. Kao, Eds., Springer International Publishing, Cham, pp. 40–52, 2014.

[14] E. SALVEMINI, F. FUMAROLA, D. MALERBA and J. HAN, *FAST sequence mining based on sparse id-lists*, in Foundations of Intelligent Systems, M. Kryszkiewicz, H. Rybinski, A. Skowron and Z. W. Ra, Eds., Springer-Verlag, Berlin, Heidelberg, pp. 316–325, 2011.

[15] J.-W. HAN, J. PEI, B. MORTAZAVI ASL, C.-M. CHEN, U. DAYAL and M.-C. HSU, *FreeSpan: frequent pattern-projected sequential pattern mining*, Proceedings of 6^{th} ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Boston, MA pp. 355–359, 2000.

[16] J. PEI, J.-W. HAN, B. MORTAZAVI ASL, J.-Y. WANG, J. PINTO, C.-M. CHEN, U. DAYAL and M.-C. HSU, *Mining sequential patterns by pattern-growth: the PrefixSpan approach*, IEEE Transactions on Knowledge and Data Engineering **16**(11), pp. 1424–1440, 2004.

[17] J. H. CHANG, *Mining weighted sequential patterns in a sequence database with a time-interval weight*, Knowledge-Based Systems **24**(1), pp. 1–9, 2011.

[18] R. A. GARCA-HERNNDEZ, J. F. MARTNEZ-TRINIDAD and J. A. CARRASCO-OCHOA, *A new algorithm for fast discovery of maximal sequential patterns in a document collection*, in Computational Linguistics and Intelligent Text Processing, A. Gelbukh, Ed., Springer-Verlag, Berlin, Heidelberg, pp. 514–523, 2006.

[19] E.-Z. GUAN, X.-Y. CHANG, Z. WANG and C.-G. ZHOU, *Mining maximal sequential patterns*, Proceedings of 2005 International Conference on Neural Networks and Brain, Beijing, China **1**, pp. 525–528, 2005.

[20] S. ZIDA, P. FOURNIER-VIGER, C.-W. WU, J. C.-W. LIN and V. S. TSENG, *Efficient mining of high-utility sequential rules*, in Machine Learning and Data Mining in Pattern Recognition, P. Perner, Ed., Springer International Publishing, Cham, pp. 157–171, 2015.

[21] F. JELINEK, J. D. LAFFERTY and R. L. MERCER, *Basic methods of probabilistic context free grammars*, in Speech Recognition and Understanding, P. Laface and R. De Mori, Eds., Springer-Verlag, Berlin, Heidelberg, pp. 345–360, 1992.

[22] T. WATANABE and E. SUMITA, *Transition-based neural constituent parsing*, Proceedings of 53^{rd} Annual Meeting of the Association for Computational Linguistics and 7^{th} International Joint Conference on Natural Language Processing, Beijing, China **1**, pp. 1169–1179, 2015.

[23] J. LIU and Y. ZHANG, *In-order transition-based constituent parsing*, Transactions of the Association for Computational Linguistics **5**, pp. 413–424, 2017.

[24] M. STERN, J. ANDREAS and D. KLEIN, *A minimal span-based neural constituency parser*, arXiv:1705.03919, 2017.

[25] O. VINYALS, . KAISER, T. KOO, S. PETROV, I. SUTSKEVER and G. HINTON, *Grammar as a foreign language*, Advances in Neural Information Processing Systems **28**, pp. 2773–2781 2015.

[26] K. S. TAI, R. SOCHER and C. D. MANNING, *Improved semantic representations from tree-structured long short-term memory networks*, arXiv:1503.00075, 2015.

[27] J.-G. BAi, Y.-J. WANG, Y.-R. CHEN, Y.-M. YANG, J. BAI, J. YU and Y.-H. TONG, *Syntax-BERT: Improving pre-trained transformers with syntax trees*, arXiv:2103.04350, 2021.

[28] A. ABEILL, K. BISHOP, S. COTE and Y. SCHABES, *A lexicalized tree adjoining grammar for English*, Technical Reports (CIS), University of Pennsylvania, Philadelphia, PA, 1990.

[29] C. NEIDLE, *Lexical functional grammar*, Encyclopedia of Language and Linguistics **5**, pp. 2147–2153, 1994.

[30] M. STEEDMAN and J. BALDRIDGE, *Combinatory categorial grammar*, in Non-Transformational Syntax: Formal and Explicit Models of Grammar, Wiley-Blackwell, pp. 181–224, 2011.

[31] D. KLEIN and C. D. MANNING, *Accurate unlexicalized parsing*, in Proceedings of 41^{st} Annual Meeting of the Association for Computational Linguistics, Sapporo, Japan, pp. 423–430, 2003.