Dependency Based Chinese Sentence Realization

Wei He[1], **Haifeng Wang**[2], **Yuqing Guo**[2], **Ting Liu**[1]
[1]Information Retrieval Lab, Harbin Institute of Technology, Harbin, China
{whe,tliu}@ir.hit.edu.cn
[2]Toshiba (China) Research and Development Center, Beijing, China
{wanghaifeng,guoyuqing}@rdc.toshiba.com.cn

Abstract

This paper describes log-linear models for a general-purpose sentence realizer based on dependency structures. Unlike traditional realizers using grammar rules, our method realizes sentences by linearizing dependency relations directly in two steps. First, the relative order between head and each dependent is determined by their dependency relation. Then the best linearizations compatible with the relative order are selected by log-linear models. The log-linear models incorporate three types of feature functions, including dependency relations, surface words and headwords. Our approach to sentence realization provides simplicity, efficiency and competitive accuracy. Trained on 8,975 dependency structures of a Chinese Dependency Treebank, the realizer achieves a BLEU score of 0.8874.

1 Introduction

Sentence realization can be described as the process of converting the semantic and syntactic representation of a sentence or series of sentences into meaningful, grammatically correct and fluent text of a particular language.

Most previous general-purpose realization systems are developed via the application of a set of grammar rules based on particular linguistic theories, e.g. Lexical Functional Grammar (LFG), Head Driven Phrase Structure Grammar (HPSG), Combinatory Categorical Grammar (CCG), Tree Adjoining Grammar (TAG) etc. The grammar rules are either developed by hand, such as those used in LinGo (Carroll et al., 1999), OpenCCG (White, 2004) and XLE (Crouch et al., 2007), or extracted automatically from annotated corpora, like the HPSG (Nakanishi et al., 2005), LFG (Cahill and van Genabith, 2006; Hogan et al., 2007) and CCG (White et al., 2007) resources derived from the Penn-II Treebank.

Over the last decade, there has been a lot of interest in a generate-and-select paradigm for surface realization. The paradigm is characterized by a separation between realization and selection, in which rule-based methods are used to generate a space of possible paraphrases, and statistical methods are used to select the most likely realization from the space. Usually, two statistical models are used to rank the output candidates. One is n-gram model over different units, such as word-level bigram/trigram models (Bangalore and Rambow, 2000; Langkilde, 2000), or factored language models integrated with syntactic tags (White et al. 2007). The other is log-linear model with different syntactic and semantic features (Velldal and Oepen, 2005; Nakanishi et al., 2005; Cahill et al., 2007).

However, little work has been done on probabilistic models learning direct mapping from input to surface strings, without the effort to construct a grammar. Guo et al. (2008) develop a general-purpose realizer couched in the framework of Lexical Functional Grammar based on simple n-gram models. Wan et al. (2009) present a dependency-spanning tree algorithm for word ordering, which first builds dependency trees to decide linear precedence between heads and modifiers then uses an n-gram language model to order siblings. Compared with n-gram model, log-linear model is more powerful in that it is easy to integrate a variety of features, and to tune feature weights to maximize the probability. A few papers have presented maximum entropy models for word or phrase ordering (Ratnaparkhi, 2000; Filippova and Strube, 2007). However, those attempts have been limited to specialized applications, such as air travel reservation or ordering constituents of a main clause in German.

This paper presents a general-purpose realizer based on log-linear models for directly linearizing dependency relations given dependency structures. We reduce the generation space by

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 809–816,
Suntec, Singapore, 2-7 August 2009.

two techniques: the first is dividing the entire dependency tree into one-depth sub-trees and solving linearization in sub-trees; the second is the determination of relative positions between dependents and heads according to dependency relations. Then the best linearization for each sub-tree is selected by the log-linear model that incorporates three types of feature functions, including dependency relations, surface words and headwords. The evaluation shows that our realizer achieves competitive generation accuracy.

The paper is structured as follows. In Section 2, we describe the idea of dividing the realization procedure for an entire dependency tree into a series of sub-procedures for sub-trees. We describe how to determine the relative positions between dependents and heads according to dependency relations in Section 3. Section 4 gives details of the log-linear model and the feature functions used for sentence realization. Section 5 explains the experiments and provides the results.

2 Sentence Realization from Dependency Structure

2.1 The Dependency Input

The input to our sentence realizer is a dependency structure as represented in the HIT Chinese Dependency Treebank (HIT-CDT)[1]. In our dependency tree representations, dependency relations are represented as arcs pointing from a head to a dependent. The types of dependency arcs indicate the semantic or grammatical relationships between the heads and the dependents, which are recorded in the dependent nodes. Figure 1 gives an example of dependency tree representation for the sentence:

(1) 这 是 武汉 航空
this is Wuhan Airline
首次 购买 波音 客机
first time buy Boeing airliner
'This is the first time for Airline Wuhan to buy Boeing airliners.'

In a dependency structure, dependents are unordered, i.e. the string position of each node is not recorded in the representation. Our sentence realizer takes such an unordered dependency tree as input, determines the linear order of the words

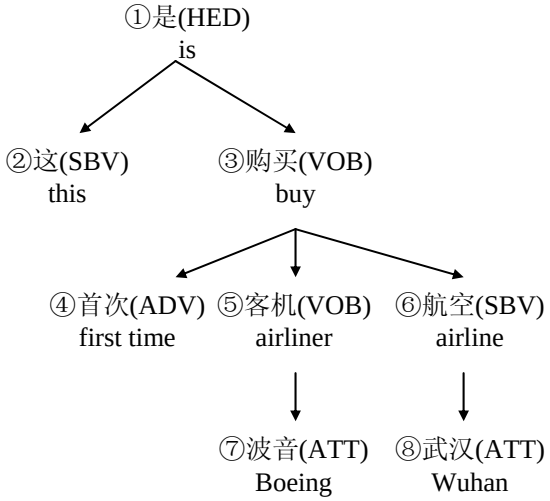

Figure 1: The dependency tree for the sentence "这是武汉航空首次购买波音客机"

as encoded in the nodes of the dependency structure and produces a grammatical sentence. As the dependency structures input to our realizer have been lexicalized, lexical selection is not involved during the surface realization.

2.2 Divide and Conquer Strategy for Linearization

For determining the linear order of words represented by nodes of the given dependency structure, in principle, the sentence realizer has to produce all possible sequences of the nodes from the input tree and selects the most likely linearization among them. If the dependency tree consists of a considerable number of nodes, this procedure would be very time-consuming. To reduce the number of possible realizations, our generation algorithm adopts a divide-and-conquer strategy, which divides the whole tree into a set of sub-trees of depth one and recursively linearizes the sub-trees in a bottom-up fashion. As illustrated in Figure 2, sub-trees *c* and *d*, which are at the bottom of the tree, are linearized first, then sub-tree *b* is processed, and finally sub-tree *a*.

The procedure imposes a projective constraint on the dependency structures, viz. each head dominates a continuous substring of the sentence realization. This assumption is feasible in the application of the dependency-based generation, because: (i) it has long been observed that the dependency structures of a vast majority of sentences in the languages of the world are projective (Igor, 1988) and (ii) non-projective dependencies in Chinese, for the most part, are used to account for non-local dependency phenomena.

[1] HIT-CDT (http://ir.hit.edu.cn) includes 10,000 sentences and 215,334 words, which are manually annotated with part-of-speech tags and dependency labels. (Liu et al., 2006a)

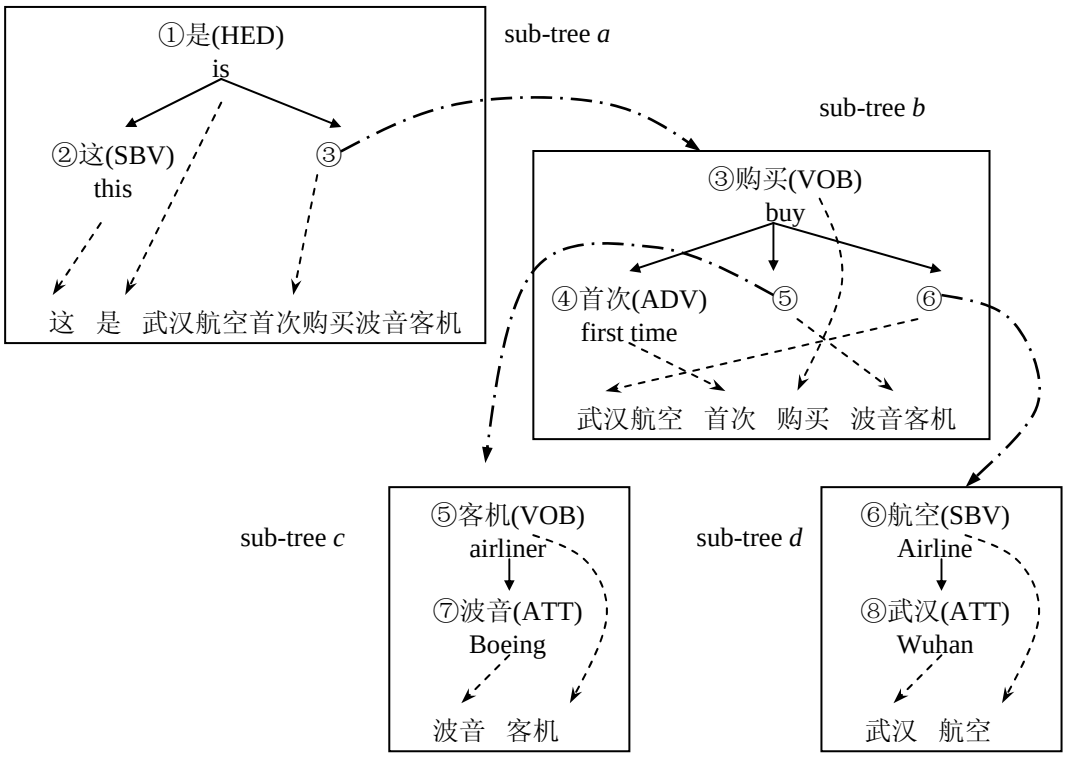

Figure 2: Illustration of the linearization procedure

Relation	Description	Postdep.	Predep.
ADV	adverbial	1	25977
APP	appositive	807	0
ATT	attribute	0	47040
CMP	complement	2931	3
CNJ	conjunctive	0	2124
COO	coordinate	6818	0
DC	dep. clause	197	0
DE	DE phrase	0	10973
DEI	DEI phrase	131	3
DI	DI phrase	0	400
IC	indep.clause	3230	0
IS	indep.structure	125	794
LAD	left adjunct	0	2644
MT	mood-tense	3203	0
POB	prep-obj	7513	0
QUN	quantity	0	6092
RAD	right adjunct	1332	1
SBV	subject-verb	6	16016
SIM	similarity	0	44
VOB	verb-object	23487	21
VV	verb-verb	6570	2

Table 1: Numbers of pre/post-dependents for each dependency relation

Though non-local dependencies are important for accurate semantic analysis, they can be easily converted to local dependencies conforming to the projective constraint. In fact, we find that the 10, 000 manually-build dependency trees of the HIT-CDT do not contain any non-projective dependencies.

3 Relative Position Determination

In dependency structures, the semantic or grammatical roles of the nodes are indicated by types of dependency relations. For example, the VOB dependency relation, which stands for the *verb-object structure*, means that the head is a verb and the dependent is an object of the verb; the ATT relation, means that the dependent is an attribute of the head. In languages with fairly rigid word order, the relative position between the head and dependent of a certain relation is in a fixed order. For example in Chinese, the object almost always occurs behind its dominating verb; the attribute modifier always occurs in front of its head word. Therefore, we can draw a conclusion that the relative positions between head and dependent of VOB and ATT can be determined by the types of dependency relations.

We make a statistic on the relative positions between head and dependent for each dependency relation type. Following (Covington, 2001), we call a dependent that precedes its head *predependent*, a dependent that follows its head *postdependent*. The corpus used to gather appropriate statistics is HIT-CDT. Table 1 gives the numbers

of predependent/postdependent for each type of dependency relations and its descriptions.

Table 1 shows that 100% dependents of ATT relation are predependents and 23,487(99.9%) against 21(0.1%) VOB dependents are postdependents. Almost all the dependency relations have a dominant dependent type—predependent or postdependent. Although some dependency relations have exceptional cases (e.g. VOB), the number is so small that it can be ignored. The only exception is the IS relation, which has 794(86.4%) predependents and 125(13.6%) postdependents. The IS label is an abbreviation for *independent structure*. This type of dependency relation is usually used to represent interjections or comments set off by brackets, which usually has little grammatical connection with the head. Figure 3 gives an example of *independent structure*. This example is from a news report, and the phrase "新华社消息" (set apart by brackets in the original text) is a supplementary explanation for the source of the news. The connection between this phrase and the main clause is so weak that either it precedes or follows the head verb is acceptable in grammar. However, this kind of news-source-explanation is customary to place at the beginning of a sentence in Chinese. This can probably explain the majority of the IS-tagged dependents are predependents.

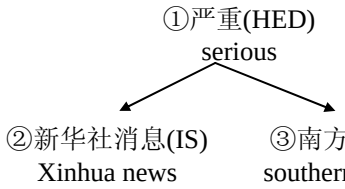

Figure 3: Example of independent structure

If we simply treat all the IS dependents as predependents, we can assume that every dependency relation has only one type of dependent, either predependent or postdependent. Therefore, the relative position between head and dependent can be determined just by the types of dependency relations.

In the light of this assumption, all dependents in a sub-tree can be classified into two groups—predependents and postdependents. The predependents must precede the head, and the postdependents must follow the head. This classification not only reduces the number of possible sequences, but also solves the linearization of a sub-tree if the sub-tree contains only one dependent, or two dependents of different types, viz. one predependent and one postdependent. In sub-tree *c* of Figure 2, the dependency relation between the only dependent and the head is ATT, which indicates that the dependent is a predependent. Therefore, node 7 is bound to precede node 5, and the only linearization result is "武汉航空". In sub-tree *a* of the same figure, the classification for SBV is predependent, and for VOB is postdependent, so the only linearization is <node 2, node 1, node 3>.

In HIT-CDT, there are 108,086 sub-trees in the 10,000 sentences, 65% sub-trees have only one dependent, and 7% sub-trees have two dependents of different types (one predependent and one postdependent). This means that the relative position classification can deterministically linearize 72% sub-trees, and only the rest 28% sub-trees with more than one predependent or postdependent need to be further determined.

4 Log-linear Models

We use log-linear models for selecting the sequence with the highest probability from all the possible linearizations of a sub-tree.

4.1 The Log-linear Model

Log-linear models employ a set of feature functions to describe properties of the data, and a set of learned weights to determine the contribution of each feature. In this framework, we have a set of M feature functions $h_m(r,t), m = 1,...,M$. For each feature function, there exists a model parameter $\lambda_m(r,t), m = 1,...,M$ that is fitted to optimize the likelihood of the training data. A conditional log-linear model for the probability of a realization *r* given the dependency tree *t*, has the general parametric form

$$p_\lambda(r \mid t) = \frac{1}{Z_\lambda(t)} \exp[\sum_{m=1}^{M} \lambda_m h_m(r,t)] \qquad (1)$$

where $Z_\lambda(t)$ is a normalization factor defined as

$$Z_\lambda(t) = \sum_{r' \in Y(t)} \exp[\sum_{m=1}^{M} \lambda_m h_m(r',t)] \qquad (2)$$

And Y(*t*) gives the set of all possible realizations of the dependency tree *t*.

4.2 Feature Functions

We use three types of feature functions for capturing relations among nodes on the dependency tree. In order to better illustrate the feature functions used in the log-linear model, we redraw sub-tree *b* of Figure 2 in Figure 4. Here we assume the linearizations of sub-tree *c* and *d* have

Feature function	Examples of features
Dependency Relation	“SBV ADV VOB” “ADV VOB VOB”
Word Model	“武汉航空首次” “航空首次购买” “首次购买波音” “购买波音客机”
Headword Model	“航空首次” “首次购买” “购买客机”

Table 2: Examples of feature functions

been finished, and the strings of linearizing results are recorded in nodes 5 and 6.

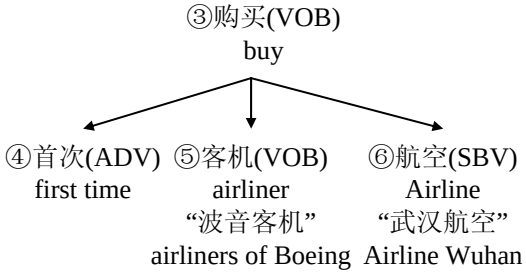

Figure 4: Sub-tree with multiple predependents

The sub-tree in Figure 4 has two predependents (SBV and ADV) and one postdependent (VOB). As a result of this classification, the only two possible linearizations of the sub-tree are <node 4, node 6, node 3, node 5> and <node 6, node 4, node 3, node 5>. Then the log-linear model that incorporates three types of feature functions is used to make further selection.

Dependency Relation Model: For a particular sub-tree structure, the task of generating a string covered by the nodes on the sub-tree is equivalent to linearizing all the dependency relations in that sub-tree. We linearize the dependency relations by computing n-gram models, similar to traditional word-based language models, except using the names of dependency relations instead of words. For the two linearizations of Figure 4, the corresponding dependency relation sequences are “ADV SBV VOB VOB” and “SBV ADV VOB VOB”. The dependency relation model calculates the probability of dependency relation n-gram $P(DR)$ according to Eq.(3). The probability score is integrated into the log-linear model as a feature.

$$P(DR_1^m) = P(DR_1 ... DR_m) \quad (3)$$
$$= \prod_{k=1}^{m} P(DR_k \mid DR_{k-n+1}^{k-1})$$

Word Model: We integrate an n-gram word model into the log-linear model for capturing the relation between adjacent words. For a string of words generated from a possible sequence of sub-tree nodes, the word models calculate word-based n-gram probabilities of the string. For example, in Figure 4, the strings generated by the two possible sequences are “武汉航空 首次 购买 波音客机” and “首次 武汉航空 购买 波音客机”. The word model takes these two strings as input, and calculates the n-gram probabilities.

Headword Model: [2] In dependency representations, heads usually play more important roles than dependents. The headword model calculates the n-gram probabilities of headwords, without regard to the words occurring at dependent nodes, in that dependent words are usually less important than headwords. In Figure 4, the two possible sequences of headwords are “航空 首次 购买 客机” and “首次 航空 购买 客机”. The headword strings are usually more generic than the strings including all words, and thus the headword model is more likely to relax the data sparseness.

Table 2 gives some examples of all the features used in the log-linear model. The examples listed in the table are features of the linearization <node 6, node 4, node 3, node 5>, extracted from the sub-tree in Figure 4.

In this paper, all the feature functions used in the log-linear model are n-gram probabilities. However, the log-linear framework has great potential for including other types of features.

4.3 Parameter Estimation

BLEU score, a method originally proposed to automatically evaluate machine translation quality (Papineni et al., 2002), has been widely used as a metric to evaluate general-purpose sentence generation (Langkilde, 2002; White et al., 2007; Guo et al. 2008, Wan et al. 2009). The BLEU measure computes the geometric mean of the precision of n-grams of various lengths between a sentence realization and a (set of) reference(s).

To estimate the parameters $(\lambda_1,...,\lambda_M)$ for the feature functions $(h_1,...,h_M)$, we use BLEU[3] as optimization objective function and adopt the approach of minimum error rate training

[2] Here the term “headword” is used to describe the word that occurs at head nodes in dependency trees.

[3] The BLEU scoring script is supplied by NIST Open Machine Translation Evaluation at ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl

	Model	BLEU	ExMatch	SSA
1	Random	0.1478	0.0038	0.2044
2	RPD + Random	0.5943	0.1274	0.6369
3	RPD + DR	0.7204	0.2167	0.7683
4	RPD + Bi-WM	0.8289	0.4125	0.8270
5	RPD + Tri-WM	0.8508	0.4715	0.8415
6	RPD + HW	0.7592	0.2909	0.7638
7	RPD + DR + Bi-WM	0.8615	0.4810	0.8723
8	RPD + DR + Tri-WM	0.8772	0.5247	0.8817
9	RPD + DR + Tri-WM + HW	**0.8874**	**0.5475**	**0.8920**

Table 3: BLEU, ExMatch and SSA scores on the test set

(MERT), which is popular in statistical machine translation (Och, 2003).

4.4 The Realization Algorithm

The realization algorithm is a recursive procedure that starts from the root node of the dependency tree, and traverses the tree by depth-first search. The pseudo code of the realization algorithm is shown in Figure 5.

```
1:procedure SEARCH
2:input: sub-tree T {head:H dep.:D1…Dn}
3:   if n = 0 then return
4:   for i := 1 to n
5:     SEARCH(Di)
6:   Apre := {}
7:   Apost := {}
8:   for i := 1 to n
9:     if PRE-DEP(Di) then Apre:=Apre∪{Di}
10:    if POST-DEP(Di) then Apost:=Apost∪{Di}
11: for all permutations p1 of Apre
12:    for all permutations p2 of Apost
13:      sequence s := JOIN(p1,H,p2)
14:      score r := LOG-LINEAR(s)
15:      if best-score(r) then RECORD(r,s)
```

Figure 5: The algorithm for linearizations of subtrees

5 Experiments

5.1 Experimental Design

Our experiments are carried out on HIT-CDT. We randomly select 526 sentences as the test set, and 499 sentences as the development set for optimizing the model parameters. The rest 8,975 sentences of the HIT-CDT are used for training of the dependency relation model. For training of word models, we use the Xinhua News part (6,879,644 words) of Chinese Gigaword Second Edition (LDC2005T14), segmented by the Language Technology Platform (LTP)[4]. And for training the headword model, we use both the HIT-CDT and the HIT Chinese Skeletal Dependency Treebank (HIT-CSDT). HIT-CSDT is a component of LTP and contains 49,991 sentences in dependency structure representation (without dependency relation labels).

As the input dependency representation does not contain punctuation information, we simply remove all punctuation marks in the test and development sets.

5.2 Evaluation Metrics

In addition to BLEU score, percentage of exactly matched sentences and average NIST simple string accuracy (SSA) are adopted as evaluation metrics. The exact match measure is percentage of the generated string that exactly matches the corresponding reference sentence. The average NIST simple string accuracy score reflects the average number of insertion (I), deletion (D), and substitution (S) errors between the output sentence and the reference sentence. Formally, SSA = $1 - (I + D + S) / R$, where R is the number of tokens in the reference sentence.

5.3 Experimental Results

All the evaluation results are shown in Table 3. The first experiment, which is a baseline experiment, ignores the tree structure and randomly chooses position for every word. From the second experiment, we begin to utilize the tree structure and apply the realization algorithm described in Section 4.4. In the second experiment, predependents are distinguished from postdependents by the relative position determination method (RPD), then the orders inside predependents and postdependents are chosen randomly. From the third experiments, the log-linear models are used for scoring the generated sequences, with the aid of three types of feature functions as described in Section 4.2. First, the feature functions of trigram dependency relation model (DR), bigram word model (Bi-WM), trigram word model (Tri-WM) (with Katz backoff) and trigram headword model (HW) are used separately in experiments 3-6. Then we combine the feature

[4] http://ir.hit.edu.cn/demo/ltp

functions incrementally based on the RPD and DR model.

The relative position determination plays an important role in the realization algorithm. We observe that the BLEU score is boosted from 0.1478 to 0.5943 by using the RPD method. This can be explained by the reason that the linearizations of 72% sub-trees can be definitely determined by the RPD method. All of the four feature functions we have tested achieve considerable improvement in BLEU scores. The dependency relation model achieves 0.7204, the bigram word model 0.8289, the trigram word model 0.8508 and the headword model achieves 0.7592. While the combined models perform better than any of their individual component models. On the foundation of relative position determination method, the combination of dependency relation and bigram word model achieves a BLEU score of 0.8615, and the combination of dependency relation and trigram word model achieves a BLEU score of 0.8772. Finally the combination of dependency relation model, trigram word model and headword model achieves the best result 0.8874.

5.4 Discussion

We first inspected the errors made by the relative position determination method. In the treebank-tree test set, there are 7 predependents classified as postdependents and 3 postdependents classified as predependents by error. Among the 9,384 dependents, the error rate of the relative position determination method is very small (0.1%).

Then we make a classification on the errors in the experiment of dependency relation model (with relative position determination method). Table 4 shows the distribution of the errors.

The first type of errors is caused by duplicate dependency relations, i.e. a head with two or more dependents that have the same dependency relations. In this situation, only using the dependency relation model cannot generate the right linearization. However, word models, which utilize the word information, can make distinctions between the dependencies. The reason for the errors of SBV-ADV and ATT-QUN is probably because the order of these pairs of grammar roles

	Error types	Proportion
1	Duplicate dependency relations	60.0%
2	SBV-ADV	20.3%
3	ATT-QUN	6.3%
4	Other	13.4%

Table 4: Error types in the RPD+DR experiment

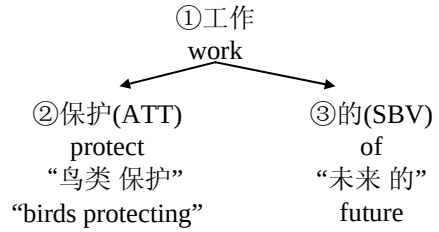

Figure 6: Sub-tree for “未来的鸟类保护工作”

is somewhat flexible. For example, the strings of “今天(ADV)/today 我(SBV)/I” and “我(SBV)/I 今天(ADV)/today” are both very common and acceptable in Chinese.

The word models tend to combine the nodes that have strong correlation together. For example in Figure 6, node 2 is more likely to precede node 3 because the words “保护/protect” and “未来/future” have strong correlation, but the correct order is <node 3, node 2>.

Headword model only consider the words occur at head nodes, which is helpful in the situation like Figure 6. In our experiments, the headword model gets a relatively low performance by itself, however, the addition of headword model to the combination of the other two feature functions improves the result from 0.8772 to 0.8874. This indicates that the headword model is complementary to the other feature functions.

6 Conclusions

We have presented a general-purpose realizer based on log-linear models, which directly maps dependency relations into surface strings. The linearization of a whole dependency tree is divided into a series of sub-procedures on sub-trees. The dependents in the sub-trees are classified into two groups, predependents or postdependents, according to their dependency relations. The evaluation shows that this relative position determination method achieves a considerable result. The log-linear model, which incorporates three types of feature functions, including dependency relation, surface words and headwords, successfully captures factors in sentence realization and demonstrates competitive performance.

References

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a Probabilistic Hierarchical Model for Generation. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 42-48. Saarbrücken, Germany.

Aoife Cahill and Josef van Genabith. 2006. Robust PCFG-Based Generation Using Automatically Acquired LFG Approximations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1033-1040. Sydney, Australia.

Aoife Cahill, Martin Forst and Christian Rohrer. 2007. Stochastic Realisation Ranking for a Free Word Order language. In *Proceedings of 11th European Workshop on Natural Language Generation*, pages 17-24. Schloss Dagstuhl, Germany.

John Carroll, Ann Copestake, Dan Flickinger, and Victor Poznanski. 1999. An Efficient Chart Generator for (Semi-)Lexicalist Grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation*, pages 86-95, Toulouse.

Michael A. Covington. 2001. A Fundamental Algorithm for Dependency Parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*, pages 95–102.

Dick Crouch, Mary Dalrymple, Ron Kaplan, Tracy King, John Maxwell, and Paula Newman. 2007. XLE documentation. Palo Alto Research Center, CA.

Katja Filippova and Michael Strube. 2007. Generating Constituent Order in German Clauses. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 320-327. Prague, Czech Republic.

Yuqing Guo, Haifeng Wang and Josef van Genabith. 2008. Dependency-Based N-Gram Models for General Purpose Sentence Realisation. In *Proceedings of the 22th International Conference on Computational Linguistics*, pages 297-304. Manchester, UK.

Deirdre Hogan, Conor Cafferkey, Aoife Cahill and Josef van Genabith. 2007. Exploiting Multi-Word Units in History-Based Probabilistic Generation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and CoNLL*, pages 267-276. Prague, Czech Republic.

Mel'čuk Igor. 1988. Dependency syntax: Theory and practice. In *Suny Series in Linguistics*. State University of New York Press, New York, USA.

Irene Langkilde. 2000. Forest-Based Statistical Sentence Generation. In *Proceedings of 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 170-177. Seattle, WA.

Irene Langkilde. 2002. An Empirical Verification of Coverage and Correctness for a General-Purpose Sentence Generator. In *Proceedings of the Second International Conference on Natural Language Generation*, pages 17-24. New York, USA.

Ting Liu, Jinshan Ma, and Sheng Li. 2006a. Building a Dependency Treebank for Improving Chinese Parser. *Journal of Chinese Language and Computing*, 16(4): 207-224.

Ting Liu, Jinshan Ma, Huijia Zhu, and Sheng Li. 2006b. Dependency Parsing Based on Dynamic Local Optimization. In *Proceedings of CoNLL-X*, pages 211-215, New York, USA.

Hiroko Nakanishi, Yusuke Miyao and Jun'ichi Tsujii. 2005. Probabilistic Models for Disambiguation of an HPSG-Based Chart Generator. In *Proceedings of the 9th International Workshop on Parsing Technology*, pages 93-102. Vancouver, British Columbia.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160-167, Sapporo, Japan.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311-318. Philadelphia, PA.

Adwait Ratnaparkhi. 2000. Trainable Methods for Natural Language Generation. In *Proceedings of North American Chapter of the Association for Computational Linguistics*, pages 194-201. Seattle, WA.

Erik Velldal and Stephan Oepen. 2005. Maximum Entropy Models for Realization Ranking. In *Proceedings of the 10th Machine Translation Summit*, pages 109-116. Phuket, Thailand,

Stephen Wan, Mark Dras, Robert Dale, Cécile Paris. 2009. Improving Grammaticality in Statistical Sentence Generation: Introducing a Dependency Spanning Tree Algorithm with an Argument Satisfaction Model. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 852-860. Athens, Greece.

Michael White. 2004. Reining in CCG Chart Realization. In *Proceedings of the third International Natural Language Generation Conference*, pages 182-191. Hampshire, UK.

Michael White, Rajakrishnan Rajkumar and Scott Martin. 2007. Towards Broad Coverage Surface Realization with CCG. In *Proceedings of the Machine Translation Summit XI Workshop*, pages 22-30. Copenhagen, Danmark.