Coordinate Structure Analysis with Global Structural Constraints and Alignment-Based Local Features

Kazuo Hara Masashi Shimbo Hideharu Okuma Yuji Matsumoto
Graduate School of Information Science
Nara Institute of Science and Technology
Ikoma, Nara 630-0192, Japan
{kazuo-h,shimbo,hideharu-o,matsu}@is.naist.jp

Abstract

We propose a hybrid approach to coordinate structure analysis that combines a simple grammar to ensure consistent global structure of coordinations in a sentence, and features based on sequence alignment to capture local symmetry of conjuncts. The weight of the alignment-based features, which in turn determines the score of coordinate structures, is optimized by perceptron training on a given corpus. A bottom-up chart parsing algorithm efficiently finds the best scoring structure, taking both nested or non-overlapping flat coordinations into account. We demonstrate that our approach outperforms existing parsers in coordination scope detection on the Genia corpus.

1 Introduction

Coordinate structures are common in life science literature. In Genia Treebank Beta (Kim et al., 2003), the number of coordinate structures is nearly equal to that of sentences. In clinical papers, the outcome of clinical trials is typically described with coordination, as in

> Median times to progression and median survival times were 6.1 months and 8.9 months in arm A and 7.2 months and 9.5 months in arm B. (Schuette et al., 2006)

Despite the frequency and implied importance of coordinate structures, coordination disambiguation remains a difficult problem even for state-of-the-art parsers. Figure 1(a) shows the coordinate structure extracted from the output of Charniak and Johnson's (2005) parser on the above example. This is somewhat surprising, given that the symmetry of conjuncts in the sentence is obvious to human eyes, and its correct coordinate structure shown in Figure 1(b) can be readily observed.

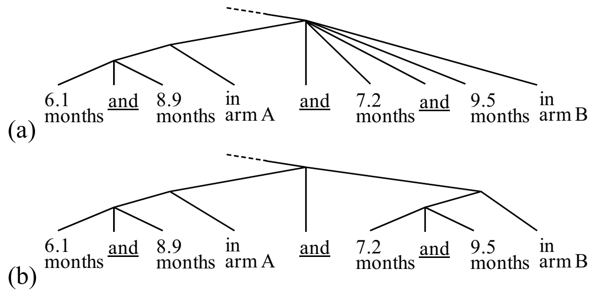


Figure 1: (a) Output from the Charniak-Johnson parser and (b) the correct coordinate structure.

Structural and semantic symmetry of conjuncts is one of the frequently observed features of coordination. This feature has been explored by previous studies on coordination, but these studies often dealt with a restricted form of coordination with apparently too much information provided from outside. Sometimes it was assumed that the coordinate structure contained two conjuncts each solely composed of a few nouns; and in many cases, the longest span of coordination (e.g., outer noun phrase scopes) was given a priori. Such rich information might be given by parsers, but this is still an unfounded assumption.

In this paper, we approach coordination by taking an extreme stance, and assume that the input is a whole sentence with no subsidiary information except for the parts-of-speech of words.

As it assumes minimal information about syntactic constructs, our method provides a baseline for future work exploiting deeper syntactic information for coordinate structure analysis. Moreover, this stand-alone approach has its own merits as well:

1. Even apart from parsing, the output coordinate structure alone may provide valuable information for higher-level applications, in the same vein as the recent success of named entity recognition and other shallow parsing

technologies. One such potential application is extracting the outcome of clinical tests as illustrated above.

2. As the system is designed independently from parsers, it can be combined with any types of parsers (e.g., phrase structure or dependency parsers), if necessary.

3. Because coordination bracketing is sometimes inconsistent with phrase structure bracketing, processing coordinations apart from phrase structures might be beneficial. Consider, for example,

   John likes, and Bill adores, Sue.
   (Carston and Blakemore, 2005)

   This kind of structure might be treated by assuming the presence of null elements, but the current parsers have limited ability to detect them. On the other hand, the symmetry of conjuncts, *John likes* and *Bill adores*, is rather obvious and should be easy to detect.

The method proposed in this paper builds a tree-like coordinate structure from the input sentence annotated with parts-of-speech. Each tree is associated with a score, which is defined in terms of features based on sequence alignment between conjuncts occurring in the tree. The feature weights are optimized with a perceptron algorithm on a training corpus annotated with the scopes of conjuncts.

The reason we build a tree of coordinations is to cope with nested coordinations, which are in fact quite common. In Genia Treebank Beta, for example, about $1/3$ of the whole coordinations are nested. The method proposed in this paper improves upon our previous work (Shimbo and Hara, 2007) which also takes a sentence as input but is restricted to flat coordinations. Our new method, on the other hand, can successfully output the correct nested structure of Figure 1(b).

2 Related work

Resnik (1999) disambiguated coordinations of the form [n_1 and n_2 n_3], where n_i are all nouns. This type of phrase has two possible readings: [(n_1) and (n_2 n_3)] and [((n_1) and (n_2)) n_3]. He demonstrated the effectiveness of semantic similarity calculated from a large text collection, and agreement of numbers between n_1 and n_2 and between n_1 and n_3. Nakov and Hearst (2005) collected web-based statistics with search engines and applied them to a task similar to Resnik's.

Hogan (2007) improved the parsing accuracy of sentences in which coordinated noun phrases are known to exist. She presented a generative model incorporating symmetry in conjunct structures and dependencies between coordinated head words. The model was then used to rerank the *n*-best outputs of the Bikel parser (2005).

Recently, Buyko et al. (2007; 2008) and Shimbo and Hara (2007) applied discriminative learning methods to coordinate structure analysis. Buyko et al. used a linear-chain CRF, whereas Shimbo and Hara proposed an approach based on perceptron learning of edit distance between conjuncts.

Shimbo and Hara's approach has its root in Kurohashi and Nagao's (1994) rule-based method for Japanese coordinations. Other studies on coordination include (Agarwal and Boggess, 1992; Chantree et al., 2005; Goldberg, 1999; Okumura and Muraki, 1994).

3 Proposed method

We propose a method for learning and detecting the scopes of coordinations. It makes no assumption about the number of coordinations in a sentence, and the sentence can contain either nested coordinations, multiple flat coordinations, or both.

The method consists of (i) a simple grammar tailored for coordinate structure, and (ii) a perceptron-based algorithm for learning feature weights. The features are defined in terms of sequence alignment between conjuncts.

We thus use the grammar to filter out inconsistent nested coordinations and non-valid (overlapping) conjunct scopes, and the alignment-based features to evaluate the similarity of conjuncts.

3.1 Grammar for coordinations

The sole objective of the grammar we present below is to ensure the consistency of two or more coordinations in a sentence; i.e., for any two coordinations, either (i) they must be totally non-overlapping (non-nested coordinations), or (ii) one coordination must be embedded within the scope of a conjunct of the other coordination (nested coordinations).

Below, we call a parse tree built from the grammar a *coordination tree*.

Table 1: Non-terminals

COORD	Complete coordination.
COORD′	Partially-built coordination.
CJT	Conjunct.
N	Non-coordination.
CC	Coordinate conjunction like "and," "or," and "but".
SEP	Connector of conjuncts other than CC: e.g., punctuations like "," and ";".
W	Any word.

Table 2: Production rules for coordination trees. (… | … | …) denotes a disjunction (matches any one of the elements). A '*' matches any word.

Rules for coordinations:	
(i)	$\mathsf{COORD}_{i,m} \longrightarrow \mathsf{CJT}_{i,j}\ \mathsf{CC}_{j+1,k-1}\ \mathsf{CJT}_{k,m}$
(ii)	$\mathsf{COORD}_{i,n} \longrightarrow \mathsf{CJT}_{i,j}\ \mathsf{SEP}_{j+1,k-1}\ \mathsf{COORD}'_{k,n}[m]$
(iii)	$\mathsf{COORD}'_{i,m}[j] \longrightarrow \mathsf{CJT}_{i,j}\ \mathsf{CC}_{j+1,k-1}\ \mathsf{CJT}_{k,m}$
(iv)	$\mathsf{COORD}'_{i,n}[j] \longrightarrow \mathsf{CJT}_{i,j}\ \mathsf{SEP}_{j+1,k-1}\ \mathsf{COORD}'_{k,n}[m]$
Rules for conjuncts:	
(v)	$\mathsf{CJT}_{i,j} \longrightarrow (\mathsf{COORD} \mid \mathsf{N})_{i,j}$
Rules for non-coordinations:	
(vi)	$\mathsf{N}_{i,k} \longrightarrow \mathsf{COORD}_{i,j}\ \mathsf{N}_{j+1,k}$
(vii)	$\mathsf{N}_{i,j} \longrightarrow \mathsf{W}_{i,i}\ (\mathsf{COORD} \mid \mathsf{N})_{i+1,j}$
(viii)	$\mathsf{N}_{i,i} \longrightarrow \mathsf{W}_{i,i}$
Rules for pre-terminals:	
(ix)	$\mathsf{CC}_{i,i} \longrightarrow (\boxed{\text{and}} \mid \boxed{\text{or}} \mid \boxed{\text{but}})_i$
(x)	$\mathsf{CC}_{i,i+1} \longrightarrow (\boxed{,} \mid \boxed{;})_i\ (\boxed{\text{and}} \mid \boxed{\text{or}} \mid \boxed{\text{but}})_{i+1}$
(xi)	$\mathsf{SEP}_{i,i} \longrightarrow (\boxed{,} \mid \boxed{;})_i$
(xii)	$\mathsf{W}_{i,i} \longrightarrow *_i$

3.1.1 Non-terminals

The grammar is composed of non-terminal symbols listed in Table 1. The distinction between COORD and COORD′ is made to cope with three or more conjuncts in a coordination. For example "*a* , *b* and *c*" is treated as a tree of the form (*a* , (*b* and *c*))), and the inner tree (*b* and *c*) is not a complete coordination, until it is conjoined with the first conjunct *a*. We represent this inner tree by a COORD′ (partial coordination), to distinguish it from a complete coordination represented by COORD. Compare Figures 2(a) and (b), which respectively depict the coordination tree for this example, and a tree for nested coordination with a similar structure.

3.1.2 Production rules

Table 2 lists the production rules. Rules are shown with explicit subscripts indicating the span of their production. The subscript to a terminal word (shown in a box) specifies its position within a sentence (word index). Non-terminals have two subscript indices denoting the span of the production.

COORD′ in rules (iii) and (iv) has an extra index *j* shown in brackets. This bracketed index maintains the end of the first conjunct (CJT) on the right-hand side. After a COORD′ is produced by these rules, it may later constitute a larger COORD or COORD′ through the application of productions (ii) or (iv). At this point, the bracketed index of the constituent COORD′ allows us to identify the scope of the first conjunct immediately underneath. As we describe in Section 3.2.4, the scope of this conjunct is necessary to compute the score of coordination trees.

These grammar rules are admittedly minimal and need further elaboration to cover all real use cases of coordination (e.g., conjunctive phrases like "as well as", etc.). Yet they are sufficient to generate the basic trees illustrated in Figure 2. The experiments of Section 5 will apply this grammar on a real biomedical corpus.

Note that although non-conjunction cue expressions, such as "both" and "either," are not the part of this grammar, such cues can be *learned* (through perceptron training) from training examples if appropriate features are introduced. Indeed, in Section 5 we use features indicating which words precede coordinations.

3.2 Score of a coordination tree

Given a sentence, our system outputs the coordination tree with the highest score among all possible trees for the sentence. The score of a coordination tree is simply the sum of the scores of all its nodes, and the node scores are computed independently from each other. Hence a bottom-up chart parsing algorithm can be designed to efficiently compute the highest scoring tree.

While scores can be assigned to any nodes, we have chosen to assign a non-zero score only to two types of *coordination nodes*, namely COORD and COORD′, in the experiment of Section 5; all other nodes are ignored in score computation. The score of a coordination node is defined via sequence alignment (Gusfield, 1997) between conjuncts below the node, to capture the symmetry of these

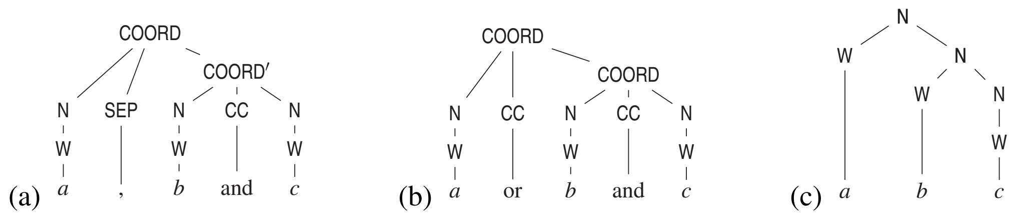


Figure 2: Coordination trees for (a) a coordination with three conjuncts, (b) nested coordinations, and (c) a non-coordination. The CJT nodes in (a) and (b) are omitted for brevity.

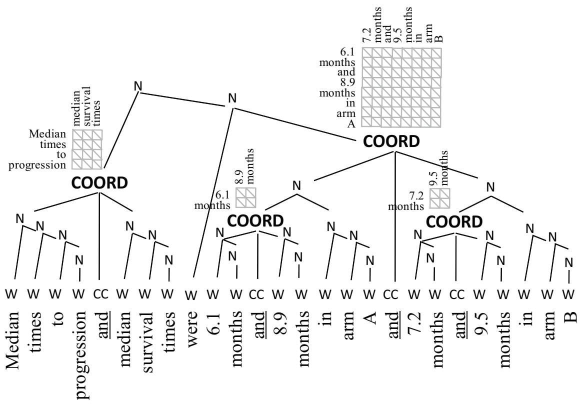


Figure 3: A coordination tree for the example sentence presented in Section 1, with the edit graphs attached to COORD nodes.

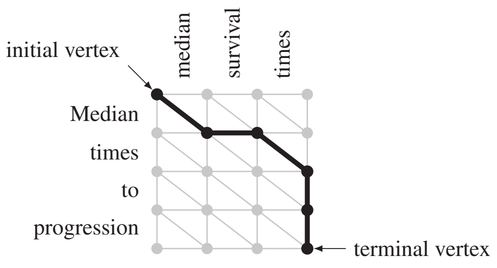


Figure 4: An edit graph and an alignment path (bold line).

conjuncts.

Figure 3 schematically illustrates the relation between a coordination tree and alignment-based computation of the coordination nodes. The score of this tree is given by the sum of the scores of the four COORD nodes, and the score of a COORD node is computed with the *edit graph* shown above the node.

3.2.1 Edit graph

The edit graph is a basic data structure for computing sequence alignment. An example edit graph is depicted in Figure 4 for word sequences "Median times to progression" and "median survival times."

A diagonal edge represents alignment (or *substitution*) between the word at the top of the edge and the one on the left, while horizontal and vertical edges represent skipping (or *deletion*) of respective word. With this representation, a path starting from the top-left corner (initial vertex) and arriving at the bottom-right corner (terminal vertex) corresponds one-to-one to a sequence of edit operations transforming one word sequence to the other.

In standard sequence alignment, each edge of an edit graph is associated with a score representing the merit of the corresponding edit operation. By defining the score of a path as the total score of its component edges, we can assess the similarity of a pair of sequences as the maximum score over all paths in its edit graph.

3.2.2 Features

In our model, instead of assigning a score independently to edges of an edit graph, we assign a vector of *features* to edges. The score of an edge is the inner product of this feature vector and another vector **w**, called *global weight vector*. Feature vectors may differ from one edge to another, but the vector **w** is unique in the entire system and consistently determines the relative importance of individual features.

In parallel to the definition of a path score, the feature vector of a path can be defined as the sum of the feature vectors assigned to its component edges. Then the score of a path is equal to the inner product $\langle \mathbf{w}, \mathbf{f} \rangle$ of **w** and the feature vector **f** of the path.

A feature assigned to an edge can be an arbitrary indicator of edge directions (horizontal, vertical, or diagonal), edge coordinates in the edit graph, attributes (such as the surface form, part-of-speech, and the location in the sentence) of the current or surrounding words, or their combination. Section 5.3 will describe the exact features used in our experiments.

3.2.3 Averaged path score as the score of a coordination node

Finally, we define the score of a COORD (or COORD′) node in a coordination tree as the *average* score of all paths in its associated edit graph. This is another deviation from standard sequence alignment, in that we do not take the maximum scoring paths as representing the similarity of conjuncts, but instead use the average over all paths.

Notice that the average is taken over paths, and not edges. In this way, a natural bias is incurred towards features occurring near the diagonal connecting the initial vertex and the terminal vertex. For instance, in an edit graph of size 8×8, there is only one path that goes through the vertex at the top-right corner, while more than 3,600 paths pass through the vertex at the center of the graph. In other words, the features associated with the center vertex receives 3,600 times more weights than those at the top-right corner after averaging.

The major benefit of this averaging is the reduced computation during training. During the perceptron training, the global weight vector $\mathbf{w}$ changes and the score of individual paths changes accordingly. On the other hand, the average *feature vector* $\bar{\mathbf{f}}$ (as opposed to the average *score* $\langle \mathbf{w}, \bar{\mathbf{f}} \rangle$) over all paths in the edit graph remains constant. This means that $\bar{\mathbf{f}}$ can be pre-computed once before the training starts, and the score computation during training reduces to simply taking the inner product of the current $\mathbf{w}$ and the pre-computed $\bar{\mathbf{f}}$.

Alternatively, the alignment score could be defined as that of the best scoring path with respect to the current $\mathbf{w}$, following the standard sequence alignment computation. However, it would require running the Viterbi algorithm in each iteration of the perceptron training, for all possible spans of conjuncts. While we first pursued this direction, it was abandoned as the training was intolerably slow.

3.2.4 Coordination with three or more conjuncts

For a coordination with three or more conjuncts, we define its score as the sum of the similarity scores of all pairwise consecutive conjuncts; i.e., for a coordination "*a*, *b*, *c*, and *d*" with four conjuncts, the score is the sum of the similarity scores for conjunct pairs (*a*, *b*), (*b*, *c*), and (*c*, *d*). Ideally, we should take all combinations of conjuncts into account, but it would lead to a combinatorial explosion and is impractical.

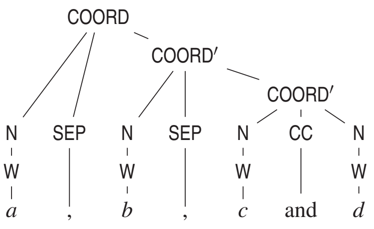


Figure 5: A coordination tree with four conjuncts. All CJT nodes are omitted.

Recall that in the grammar introduced in Section 3.1, we attached a bracketed index to COORD′. This bracketed index was introduced for the computation of this pairwise similarity.

Figure 5 shows the coordination tree for "*a*, *b*, *c*, and *d*." The pairwise similarity scores for (*a*, *b*), (*b*, *c*), and (*c*, *d*) are respectively computed at the top COORD, left COORD′, and right COORD′ nodes, using the scheme described in Section 3.2.3. To compute the similarity of *a* and *b*, we need to lift the information about the end position of *b* upward to the COORD node. The same applies to computing the similarity of *b* and *c*; the end position of *c* is needed at the left COORD′. The bracketed index of COORD′ exactly maintains this information, i.e., the end of the first conjunct below the COORD′. See production rules (iii) and (iv) in Table 2.

3.3 Perceptron learning of feature weights

As we saw above, our model is a linear model with the global weight vector $\mathbf{w}$ acting as the coefficient vector, and hence various existing techniques can be exploited to optimize $\mathbf{w}$.

In this paper, we use the averaged perceptron learning (Collins, 2002; Freund and Schapire, 1999) to optimize $\mathbf{w}$ on a training corpus, so that the system assigns the highest score to the correct coordination tree among all possible trees for each training sentence.

4 Discussion

4.1 Computational complexity

Given an input sentence of N words, finding its maximum scoring coordination tree by a bottom-up chart parsing algorithm incurs a time complexity of $O(N^3)$.

While the right-hand side of rules (i)–(iv) involves more than three variables and thus appears to increase complexity, this is not the case since some of the variables (j and k in rules (i) and (iii), and j, k, and m in rules (ii) and (iv)) are constrained by the location of conjunct connectors (CC and SEP), whose number in a sentence is negligible compared to the sentence length N. As a result, these rules can be processed in $O(N^2)$ time. Hence the run-time complexity is dominated by rule (vi), which has three variables and leads to $O(N^3)$.

Each iteration of the perceptron algorithm for a sentence of length N also incurs $O(N^3)$ for the same reason.

Our method also requires pre-processing in the beginning of perceptron training, to compute the average feature vectors $\bar{\mathbf{f}}$ for all possible spans (i, j) and (k,m) of conjuncts in a sentence. With a reasoning similar to the complexity analysis of the chart parsing algorithm above, we can show that the pre-processing takes $O(N^4)$ time.

4.2 Difference from Shimbo and Hara's method

The method proposed in this paper extends the work of Shimbo and Hara (2007). Both take a whole sentence as input and use perceptron learning, and the difference lies in how hypothesis coordination(s) are encoded as a feature vector.

Unlike our new method which constructs a tree of coordinations, Shimbo and Hara used a *chainable* partial paths (representing non-overlapping series of local alignments; see (Shimbo and Hara, 2007, Figure 5)) in a global triangular edit graph.

In our method, we compute many edit graphs of smaller size, one for each possible conjunct pair in a sentence. We use global alignment (a complete path) in these smaller graphs, as opposed to chainable local alignment (partial paths) in a global edit graph used by Shimbo and Hara.

Since nested coordinations cannot be encoded as chainable partial paths (Shimbo and Hara, 2007), their method cannot cope with nested coordinations such as those illustrated in Figure 2(b).

4.3 Integration with parsers

Charniak and Johnson (2005) reported an improved parsing accuracy by reranking n-best parse trees, using features based on similarity of coordinated phrases, among others. It should be interesting to investigate whether alignment-based features like ours can be built into their reranker, or more generally, whether the coordination scopes output by our method help improving parsing accuracy.

The combinatory categorial grammar (CCG) (Steedman, 2000) provides an account for various coordination constructs in an elegant manner, and incorporating alignment-based features into the CCG parser (Clark and Curran, 2007) is also a viable possibility.

5 Evaluation

We evaluated the performance of our method[1] on the Genia corpus (Kim et al., 2003).

5.1 Dataset

Genia Treebank Beta is a collection of Penn Treebank-like phrase structure trees for 4529 sentences from Medline abstracts.

In this corpus, each scope of coordinate structures is annotated with an explicit tag, and the conjuncts are always placed inside brackets. Not many treebanks explicitly mark the scope of conjuncts; for example, the Penn Treebank frequently omits bracketing of coordination and conjunct scopes, leaving them as a flat structure.

Genia contains a total of 4129 occurrences of COOD tags indicating coordination. These tags are further subcategorized into phrase types such as NP-COOD and VP-COOD. Among coordinations annotated with COOD tags, we selected those surrounding "and," "or," and "but." This yielded 3598 coordinations (2997, 355, and 246 for "and," "or," and "but," respectively) in 2508 sentences. These coordinations constitute nearly 90% of all coordinations in Genia, and we used them as the evaluation dataset. The length of these sentences is 30.0 words on average.

5.2 Evaluation method

We tested the proposed method in two tasks:

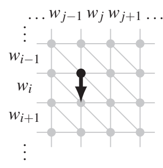
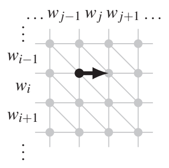
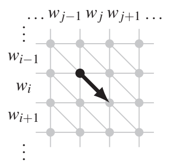
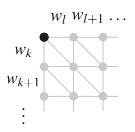
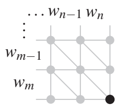
(i) identify the scope of coordinations regardless of phrase types, and

(ii) detect noun phrase (NP) coordinations and identify their scopes.

While the goal of task (i) is to determine the scopes of 3598 coordinations, task (ii) demands both to judge whether each of the coordinations constructs an NP, and if it does, to determine its scope.

[1]A C++ implementation of our method can be found at http://cl.naist.jp/project/coordination/, along with supplementary materials including the preliminary experimental results of the CCG parser on the same dataset.

Table 3: Features in the edit graph for conjuncts $w_k w_{k+1} \cdots w_m$ and $w_l w_{l+1} \cdots w_n$.

edge/vertex type	vertical edge	horizontal edge	diagonal edge	initial vertex	terminal vertex
	 $\ldots w_{j-1}\, w_j\, w_{j+1} \ldots$ w_{i-1} w_i w_{i+1}	 $\ldots w_{j-1}\, w_j\, w_{j+1} \ldots$ w_{i-1} w_i w_{i+1}	 $\ldots w_{j-1}\, w_j\, w_{j+1} \ldots$ w_{i-1} w_i w_{i+1}		
vertical bigrams	$w_{i-1}w_i$	$w_{i-1}w_i$	$w_{i-1}w_i$	$w_{k-2}w_{k-1}$	$w_{m-2}w_{m-1}$
	$w_i w_{i+1}$		$w_i w_{i+1}$	$w_{k-1}w_k$	$w_{m-1}w_m$
				$w_k w_{k+1}$	$w_m w_{m+1}$
horizontal bigrams	$w_{j-1}w_j$	$w_{j-1}w_j$	$w_{j-1}w_j$	$w_{l-2}w_{l-1}$	$w_{n-2}w_{n-1}$
		$w_j w_{j+1}$	$w_j w_{j+1}$	$w_{l-1}w_l$	$w_{n-1}w_n$
				$w_l w_{l+1}$	$w_n w_{n+1}$
orthogonal bigrams			$w_i w_j$	$w_{k-1}w_{l-1}$	$w_{m-1}w_{n-1}$
				$w_{k-1}w_l$	$w_{m-1}w_n$
				$w_k w_{l-1}$	$w_m w_{n-1}$
				$w_k w_l$	$w_m w_n$

For comparison, two parsers, the Bikel-Collins parser (Bikel, 2005)[2] and Charniak-Johnson reranking parser[3], were applied in both tasks.

Task (ii) imitates the evaluation reported by Shimbo and Hara (2007), and to compare our method with their coordination analysis method. Because their method can only process flat coordinations, in task (ii) we only used 1613 sentences in which "and" occurs just once, following (Shimbo and Hara, 2007). Note however that the split of data is different from their experiments.

We evaluate the performance of the tested methods by the accuracy of coordination-level bracketing (Shimbo and Hara, 2007); i.e., we count each of the coordination (as opposed to conjunct) scopes as one output of the system, and the system output is deemed correct if the beginning of the first output conjunct and the end of the last conjunct both match annotations in the Genia Treebank.

In both tasks, we report the micro-averaged results of five-fold cross validation.

The Bikel-Collins and Charniak-Johnson parsers were trained on Genia, using all the phrase structure trees in the corpus except the test set; i.e., the training set also contains (in addition to the four folds) $2021 (= 4129 - 2508)$ sentences which are not in the five folds. Since the two parsers were also trained on Genia, we interpret the bracketing above each conjunction in the parse tree output by them as the coordination scope output by the parsers, in accordance with how coordinations are annotated in Genia. In testing, the Bikel-Collins parser and Shimbo-Hara method were given the gold parts-of-speech (POS) of the test sentences in Genia. We trained the proposed method twice, once with the gold POS tags and once with the POS tags output by the Charniak-Johnson parser. This is because the Charniak-Johnson parser does not accept POS tags of the test sentences.

5.3 Features

To compute features for our method, each word in a sentence was represented as a list of attributes. The attributes include the surface word, part-of-speech, suffix, prefix, and the indicators of whether the word is capitalized, whether it is composed of all uppercase letters or digits, and whether it contains digits or hyphens. All features are defined as an indicator of an attribute in two words coming from either a single conjunct (either horizontal or vertical word sequences associated with the edit graph) or two conjuncts (one from the horizontal word sequence and one from the vertical sequence). We call the first type *horizontal/vertical bigrams* and the second *orthogonal bigrams*.

Table 3 summarizes the features in an edit graph for two conjuncts $(w_k w_{k+1} \cdots w_m)$ and $(w_l w_{l+1} \cdots w_n)$, where w_i denotes the *i*th word in the sentence.

As seen from the table, features are assigned to the initial and terminal vertices as well as to edges. A $w_i w_j$ in the table indicates that for each attribute (e.g., part-of-speech, etc.), an indicator function for the combination of the attribute values in w_i and w_j is assigned to the vertex or edge shown in the figure above. Note that the features

[2]http://www.cis.upenn.edu/~dbikel/software.html

[3]ftp://ftp.cs.brown.edu/pub/nlparser/reranking-parserAug06.tar.gz

Table 4: Results of Task (i). The number of coordinations of each type (#), and the recall (%) for the proposed method, Bikel-Collins parser (BC), and Charniak-Johnson parser (CJ).

		gold POS		CJ POS	
COOD	#	Proposed	BC	Proposed	CJ
Overall	3598	**61.5**	52.1	**57.5**	52.9
NP	2317	**64.2**	45.5	**62.5**	50.1
VP	465	54.2	**67.7**	42.6	**61.9**
ADJP	321	**80.4**	66.4	**76.3**	48.6
S	188	22.9	**67.0**	15.4	**63.3**
PP	167	**59.9**	53.3	53.9	**58.1**
UCP	60	**36.7**	18.3	**38.3**	26.7
SBAR	56	51.8	**85.7**	33.9	**83.9**
ADVP	21	85.7	**90.5**	85.7	**90.5**
Others	3	**66.7**	33.3	**33.3**	0.0

assigned to different types of vertex or edge are treated as distinct even if the word indices *i* and *j* are identical; i.e., all features are conditioned on edge/vertex types to which they are assigned.

5.4 Results

Task (i) Table 4 shows the results of task (i). We only list the recall score in the table, as precision (and hence F1-measure, too) was equal to recall for all methods in this task; this is not surprising given that in this data set, conjunctions "and", "or", and "but" always indicate the existence of a coordination, and all methods successfully learned this trend from the training data.

The proposed method outperformed parsers on the coordination scope identification overall. The table also indicates that our method considerably outperformed two parsers on NP-COOD, ADJP-COOD, and UCP-COOD categories, but it did not work well on VP-COOD, S-COOD, and SBAR-COOD. In contrast, the parsers performed quite well in the latter categories.

Task (ii) Table 5 lists the results of task (ii). The proposed method outperformed Shimbo-Hara method in this task, although the setting of this task is mostly identical to (Shimbo and Hara, 2007) and does not include nested coordinations. Note also that both methods use roughly equivalent features.

One reason should be that our grammar rules can strictly enforce the scope consistency of conjuncts in coordinations with three or more conjuncts. Because the Shimbo-Hara method represents such coordinations as a series of sub-paths in an edit graph which are output independently of each other without enforcing consistency, their method can produce inconsistent scopes of conjuncts in the middle.

Table 5: Results of Task (ii). Proposed method, BC: Bikel-Collins, CJ: Charniak-Johnson, SH: Shimbo-Hara.

	gold POS			CJ POS	
	Proposed	BC	SH	Proposed	CJ
Precision	**61.7**	45.6	55.9	**60.2**	49.0
Recall	**57.9**	46.1	53.7	**55.6**	46.8
F1	**59.7**	45.8	54.8	**57.8**	47.9

In fact, the advantage of the proposed method in task (ii) is noticeable especially in coordinations with three or more conjuncts; if we restrict the test set only to coordinations with three or more conjuncts, the F-measures in the proposed method and Shimbo-Hara become 53.0 and 42.3, respectively; i.e., the margin increases to 10.7 from 4.9 points.

6 Conclusion and outlook

We have proposed a method for learning and analyzing generic coordinate structures including nested coordinations. It consists of a simple grammar for coordination and perceptron learning of alignment-based features.

The method performed well overall and on coordinated noun and adjective phrases, but not on coordinated verb phrases and sentences. The latter coordination types are in fact easy for parsers, as the experimental results show.

The proposed method failing in verbal and sentential coordinations is as expected, since conjuncts in these coordinations are not necessarily similar, if they are viewed as a sequence of words. We will investigate similarity measures different from sequence alignment, to better capture the symmetry of these conjuncts.

We will also pursue integration of our method with parsers. Because they have advantages in different coordination phrase types, their integration looks promising.

Acknowledgments

We thank anonymous reviewers for helpful comments and the pointer to the combinatory categorial grammar.

References

Rajeev Agarwal and Lois Boggess. 1992. A simple but useful approach to conjunct identification. In *Proceedings of the 30th Annual Meeting of the Associa-*

tion for Computational Linguistics (ACL'92), pages 15–21.

Daniel M. Bikel. 2005. Multilingual statistical parsing engine version 0.9.9c. http://www.cis.upenn.edu/~dbikel/software.html.

Ekaterina Buyko and Udo Hahn. 2008. Are morphosyntactic features more predicative for the resolution of noun phrase coordination ambiguity than lexico-semantic similarity scores. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)*, pages 89–96, Manchester, UK.

Ekaterina Buyko, Katrin Tomanek, and Udo Hahn. 2007. Resolution of coordination ellipses in biological named entities using conditional random fields. In *Proceedings of the Pacific Association for Computational Linguistics (PACLIC'07)*, pages 163–171.

Robyn Carston and Diane Blakemore. 2005. Editorial: Introduction to coordination: syntax, semantics and pragmatics. *Lingua*, 115:353–358.

Francis Chantree, Adam Kilgarriff, Anne de Roeck, and Alistair Willis. 2005. Disambiguating coordinations using word distribution information. In *Proceedings of the Int'l Conference on Recent Advances in Natural Language Processing*, Borovets, Bulgaria.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine *n*-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 173–180, Ann Arbor, Michigan, USA.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 1–8, Philadelphia, PA, USA.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Miriam Goldberg. 1999. An unsupervised model for statistically determining coordinate phrase attachment. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 1999)*, pages 610–614, College Park, Maryland, USA.

Dan Gusfield. 1997. *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press.

Deirdre Hogan. 2007. Coordinate noun phrase disambiguation in a generative parsing model. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL 2007)*, pages 680–687, Prague, Czech Republic.

J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. 2003. GENIA corpus: a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(Suppl. 1):i180–i182.

Sadao Kurohashi and Makoto Nagao. 1994. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20:507–534.

Preslav Nakov and Marti Hearst. 2005. Using the web as an implicit training set: application to structural ambiguity resolution. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language (HLT-EMNLP 2005)*, pages 835–842, Vancouver, Canada.

Akitoshi Okumura and Kazunori Muraki. 1994. Symmetric pattern matching analysis for English coordinate structures. In *Proceedings of the Fourth Conference on Applied Natural Language Processing*, pages 41–46.

Philip Resnik. 1999. Semantic similarity in a taxonomy. *Journal of Artificial Intelligence Research*, 11:95–130.

Wolfgang Schuette, Thomas Blankenburg, Wolf Guschall, Ina Dittrich, Michael Schroeder, Hans Schweisfurth, Assaad Chemaissani, Christian Schumann, Nikolas Dickgreber, Tabea Appel, and Dieter Ukena. 2006. Multicenter randomized trial for stage iiib/iv non-small-cell lung cancer using every-3-week versus weekly paclitaxel/carboplatin. *Clinical Lung Cancer*, 7:338–343.

Masashi Shimbo and Kazuo Hara. 2007. A discriminative learning model for coordinate conjunctions. In *Proceedings of Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pages 610–619, Prague, Czech Republic.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA, USA.